Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
Dirk Hohndel, *Intel*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
Matthew Wilson, *rPath*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
Eugene Teo, *Red Hat, Inc.*
Kyle McMartin, *Red Hat, Inc.*
Jake Edge, *LWN.net*
Robyn Bergeron
Dave Boutcher, *IBM*
Mats Wichmann, *Intel*

Where Linux Kernel Documentation Hides

Rob Landley
Impact Linux
rob@landley.net

Abstract

Last year my day job was documenting the Linux kernel. It seemed like a simple task: read through the kernel's Documentation directory, fill in the holes, and make sure it all stays up to date. In reality, I found most kernel documentation lives in web pages, magazine articles, online books, blog entires, wikis, conference papers, videos, standards, man pages, list archives, commit messages, and more. The real problem is editorial: only after finding and indexing the existing documentation can we figure out whether it's up to date and what's missing.

In this talk I'll list the documentation sources I found, show my attempts at organizing them (on `http://kernel.org/doc`), and explain what would be necessary to really get this issue under control.

1 Introduction

In 2007 the Linux Foundation awarded me a fellowship to deal with the ongoing lack of good Linux kernel documentation. Unfortunately, a good problem statement isn't necessarily a good plan of attack. I spent most of the next seven months just trying to figure out what "fixing it" actually meant, and how to go about it. The results may be found at `http://kernel.org/doc`.

My first big surprise was that the kernel's Documentation/ directory is just the tip of the iceberg. Most kernel documentation lives in web pages, magazine articles, online books, blog entries, wikis, conference papers, audio and video recordings of talks, standards, man pages, list archives, commit messages, and more. The real problem isn't a LACK of documentation, it's that no human being can ever hope to read more than a tiny fraction of what's said, written, and recorded about the kernel on a daily basis. Merging all this raw data into the kernel tarball would be like trying to burn the internet to CD.

Coping with such an enormous slush pile is fundamentally an editorial task. Only after finding and indexing the existing mass of documentation can anyone figure out whether what's out there for a given topic is complete and up-to-date. You can't fill in the holes without first knowing where they are.

This paper is not an attempt to summarize seven months of reading about why UTF-8 is a good internationalization format or how to load firmware out of initramfs for a statically linked device driver. It's about turning the dark matter of kernel documentation into something you can browse.

2 The editorial task

Google is great at finding things, but it doesn't tell you what to search for. Google is not a reference work that allows one to see available topics and home in on an area of interest, moving from more general to more specific. But there's more to teaching someone English than handing them a dictionary: a good reference work is not necessarily a good tutorial. Adding a reference index to a tutorial is fairly easy, turning a reference into a tutorial is harder. But creating a reference from scratch is also easier than creating a tutorial from scratch, a reference can be little more than a collection of sorted links, while a tutorial has to make sense.

Indexing the web's Linux kernel documentation just to provide a comprehensive reference is a huge undertaking. Even keeping an index up to date *after* its creation would be a project on par with any other major kernel subsystem. But without knowing here the holes are, writing new documentation to try to fill in those holes tends to reinvent the wheel.

In the absence of an obvious place to go on the web to find the most up-to-date existing documentation, newly created documentation tends to be repetitive and overlapping. Half-hearted attempts to collate what's available into a single new comprehensive version often just

add one more variant to the pile. A well-meaning editor who isn't already an expert on a given topic probably won't create a new category killer document attracting patches instead of competition. If they didn't feel like contributing to someone else's existing document, why would the next well-meaning editor be different?

More to the point, author and editor are different jobs. Researching and writing new documentation isn't really an editorial task. An editor collects and organizes the submissions of others. A real editor spends most of their time wading through a slush pile and saying "no" to most of it, or saying "no" to things their assistant editors pass up to them.

If this sounds familiar, it's because fighting off sturgeon's law is what editors do. Open source project maintainers perform an editorial job, publishing regular editions of source code anthologies. Putting together Linux distributions is another layer of editorial filtering. Open source developers are already familiar with this process. Applying it to documentation, providing a brief summary and a pile of links to existing documents is more effective than trying to become an expert on every single topic, and has the advantage of not making the clutter any *worse*.

The other big editorial problem is keeping documentation up to date. When code is a living thing, its documentation must also be. The best documentation about the innards of the 2.4 kernel is only passably accurate about early 2.6, and in many ways the first 2.6 release has more in common with 2.4 than with 2.6.25. But the "many eyeballs" effect of open source can be diluted by having many targets. In a maze of twisty documents, all different, fixes that don't naturally funnel to a central integration and redistribution point get eaten by a grue.

3 Why won't it all fit in the kernel tarball?

It took me a while to realize the editorial nature of the kernel documentation problem, and that it was not primarily a question of new development. The obvious place to start when looking for Linux kernel documentation may be Google, but the next most obvious place is the Documentation directory in the kernel source tarball. And that comes with some enormous built-in assumptions.

The kernel tarball is the central repository for kernel source code, so it's easy to assume that Documentation/ is the central repository for all kernel documentation, or could easily be turned into such. This mistaken assumption cost me about 3 months.

First of all, the Documentation directory isn't even the only significant source of documentation within the kernel tarball itself. The kerneldoc entries in the source code (used by `make htmldocs`) and the kconfig help entries (used by the help option of `make menuconfig`) are each significant and completely separate sources of documentation. The files in Documentation/ seldom if ever refer to htmldocs or menuconfig help, and those seldom refer back to it (or to each other).

This other information cannot easily be migrated to the Documentation directory. The other sources of documentation in the kernel source are usually located near the things they document, to benefit from locality of reference. There's a reason they live where they do, as do over two dozen README files in the source code, the output of `make help`, references to IETF RFC documents in source comments, and so on.

In addition, the data formats are different. Documentation/ consists primarily of flat text files, htmldocs uses structured source code comments to generate docbook (and from that HTML or PDF output), and kconfig is in its own format which has dedicated viewer programs (such as menuconfig).

None of these is really an obvious choice for indexing the others. The flat text of Documentation/ does not lend itself to linking out the way HTML does, so at first glance htmldocs seems a better choice for an index. But the format of htmldocs is highly constrained by its origins as structured source comments; it's designed to do what it's currently doing and not much else. As a potential index, the kconfig help entries have both sets of disadvantages; they're flat text without hyperlinks and they're highly structured for a specific purpose.

On a second look, the Documentation directory seems the least bad choice for indexing the other documentation content in the kernel tarball, so it's worth a closer look here.

4 Organized based on where passing strangers put things down last

Documentation/ does not compile, give warnings, or break the build. It cannot easily be profiled, bench-

marked, or regression tested. Because of this, the normal kernel build process doesn't naturally organize it very well. Here are a few of the files in the top level Documentation directory of the 2.6.25 kernel:

- **IRQ.txt**: Introductory file answering the question, "What is an IRQ?"
- **unicode.txt**: a standards document mirrored from lanana.org.
- **stallion.txt**: documentation for an unmaintained multiport serial card, last updated in 1999. (see also sx.txt, riscom8.txt, computone.txt...)
- **unshare.txt**: just under 300 lines of documentation about a single system call, unshare(2).
- **cli-sti-removal.txt**: Guide for migrating away from cli/sti locking, circa 2.5.28.
- **feature-removal-schedule.txt**: an important, regularly updated file about ways the Linux kernel plans to break binary compatability without necessarily involving sysfs.
- **zorro.txt**: Documentation for the Amiga "Zorro" bus.
- **spinlock.txt**: A post Linus made to the linux-kernel mailing list back in 1998 about spinlocks, with some almost intelligible notes at the top about which portions of the original message are obsolete or deprecated.
- **mono.txt** and **java.txt**: instructions on how to configure BINFMT_MISC to run Microsoft and Sun's bytecode languages. (No wine.txt to do the same for Windows binaries, though.)
- **README.cycladesZ**: file containing a URL to firmware for the Cyclades-Z card. (It does not say what a Cyclades-Z card is.)
- **logo.gif** and **logo.txt**: the Tux graphic, and a URL to Larry Ewing's page on it.
- **email-clients.txt**: Notes about sending unmangled patches to the linux kernel mailing list with alpine, evolution, kmail, lotus notes, mutt, pine, sylpheed, thunderbird, and tkrat.
- **IPMI.txt**: docs about the Intelligent Management Platform Interface driver. It's over 600 lines long but links to an Intel website in the introduction because "IPMI is a big subject and I can't cover it all here!"
- **dontdiff**: data file for use with diff's "-X" option.
- **tty.txt**: This file starts "The Lockronomicon: Your guide to the ancient and twisted locking policies of the tty layer and the warped logic behind them. Beware all ye who read on."

This is a small subset of the ~140 files at the top level, and doesn't include anything in the ~75 different subdirectories for busses, architectures, foreign language translations, subsystems, and so on.

A token attempt at organizing Documentation/ can be found in the 00-INDEX files in each subdirectory, containing a one line description of each file (example: "device-mapper/ - directory with info on Device Mapper."). Some directories have this file, some don't. Some files are listed, some aren't.

00-INDEX is better than nothing, but it mirrors a filesystem hierarchy without symlinks. A file like filesystems/ramfs-rootfs-initramfs.txt belongs both in "filesystems" and in "early-userspace", but it has to pick one.

Even the perennial question "where do I start?" has at least three answers in the kernel tarball's existing documentation: the oldest and in some ways still the best is the "README" file at the top of the kernel (not in Documentation), the next oldest is Documentation/kernel-docs.txt, and the newest is Documentation/HOWTO. None of them really provide a good introduction to the kernel's source code. For that I recommend Linux Kernel 2.4 Internals (`http://www.moses.uklinux.net/patches/lki.html`), which is woefully out of date and x86-specific but still the best I've found. It is not in the kernel tarball. (Neither is `http://en.wikibooks.org/wiki/Inside_Linux_Kernel` which seems to be another unrelated attempt at doing the same thing. There are plenty more out there.)

It is possible to clean up Documentation/ (albeit a fairly large undertaking), and I pushed a few patches to this effect (which were generally greeted with a strange

combination of indifference and bikeshedding). It's also possible to convert the Documentation directory to HTML (an even larger project, of dubious value). But ultimately, there's a larger philosophical problem.

Documentation/ is based on the assumption that everything of interest will be merged into the kernel tarball. It already copies standards documents and HOWTOs with defined upstream locations, because having potentially out-of-date copies in the kernel tarball is considered superior to having a single cannonical location for this information out on the web. The philosophy of Documentation/ is the same as for code: if out of tree drivers are bad, out of tree documentation must also be bad.

This is a difficult philosophy to apply to indexing documentation that lives on the web. The web has many formats (from pdf to flash) and Documentation has one. Web content has many licenses, the kernel is GPLv2 only. How does one apply CodingStyle to Linus Torvalds' Google video about the origins of git? The kernel source tarball is currently just under 50 megabytes, the mp3 audio recordings of OLS talks just for the year 2000 total a little under 90 megabytes.

Unfortunately, the belief that the internet can or should be distilled into Documentation/ is pervasive among kernel developers. My interview process for the Linux Foundation fellowship consisted of writing Documentation/rbtree.txt. Before doing so I pointed out that there was already an excellent article on Red Black Trees in the Linux Weekly News kernel archives, and another article about it on Wikipedia. But they weren't in the kernel tarball and thus (I was told) they didn't count, so I reinvented the wheel to get the job. Three months later, I regretted adding to fragmentation.

Exporting kernel tarball docs to `http://kernel.org/doc`

The kernel-centric Documentation/ in the kernel tarball created a reciprocal problem: Not only did Documentation/ suck at indexing the web, but the web wasn't doing that great at indexing the kernel's built-in documentation either.

I personally encountered this effect in early 2007, when my Google search for ext2 filesystem format documentation which didn't bring up Documentation/filesystems/ext2.txt in the first five pages of hits. I didn't even notice that file until a month later (after all if its Google rank sucks how good can it be), because there was no cannonical uncompressed location at which to find it on the web, and things like gitweb or the most recent release tarball were too transient to work up much of a ranking for any specific version. (Similarly, there was no standard web location for the current htmldocs, despite those being HTML!)

So the first well-defined problem I needed to tackle was exporting the documentation already in the kernel tarball somewhere Google could find it. I requested a page on kernel.org, and received `http://kernel.org/doc`. I copied the kernel's Documentation/* to `http://kernel.org/doc/Documentation`, set up the `make htmldocs` tools on my laptop, and posted the results to `http://kernel.org/doc/htmldocs`. Then I created a script to periodically update this documentation from the kernel repository (`http://kernel.org/hg/linux-2.6`) and checked this script into a new mercurial repository on my website (`http://landley.net/hg/kdocs`).

Over the months that followed, I improved my export script to harvest and export much more information from the kernel source. (See `http://landley.net/hg/kdocs/file/tip/make/` for the scripts that do all this. If you check out the mercurial repository and run "make/make.sh --long" it'll try to reproduce this directory on your machine. You need mercurial, wget, pdftk, xmlto, and probably some other stuff.)

`http://kernel.org/doc/Documentation/`

The way a web server shows a directory full of files isn't very informative, so I wrote a Python script to turn the 00-INDEX files in each Documentation subdirectory into a simple HTML index. This had the unfortunate side effect of hiding files in any directory with a 00-INDEX that doesn't list everything, so I wrote the script `make/doclinkcheck.py` which compares the generated HTML indexes against the contents of the directories and shows 404 errors and extra files. I sent lots of 00-INDEX patches to linux-kernel trying to fill in some of the gaps, but as of April 2008 doclinkcheck.py shows about 650 files still improperly indexed.

`http://kernel.org/doc/htmldocs`

On the htmldocs front, the top level book index created by `make htmldocs` was unfortunate, so I had my

script create a better one. I also wrote a quick script to create "one big html file" versions of each "book", and used the old trick that if "deviceiobook.html" is the one big ("nochunks") version, "deviceiobook/" at the same location is a directory containing the many small pages ("chunks") version. The top level index lists both versions.

http://kernel.org/doc/menuconfig/

The kconfig help text is the third big source of kernel documentation, and the only human readable documentation on several topics, so I wrote `make/menuconfig2html.py` to parse the kconfig source files and produce HTML versions of the help text.

The resulting web pages organize information the same way menuconfig does. The first page selects architecture, the later pages show config symbols with one line descriptions. The symbol names link to help text extracted from the appropriate Kconfig file.

I attempted to organize the result to reduce duplication to produce a "single point of truth" for Google to find easily, and hopefully rank high. There are several index pages (since menuconfig shows different menus for different architectures), but each Kconfig file is translated to a single page of help text, and the indexes link to the same translated Kconfig files. Each HTML file is named after the source file it's generated from.

http://kernel.org/doc/rfc-linux.html

Many comments in the Linux kernel source code reference Internet Engineering Task Force Request For Comments (IETF RFC) standards documents, which live at `http://tools.ietf.org/html`. I put together a script to grep the source code for RFC mentions, and put a link to that RFC together with links to each source file that mentions it. (It seemed like a useful thing to do at the time.)

http://kernel.org/doc/readme

The kernel source contains over two dozen README files outside of the Documentation directory. My export scripts collect them together into one directory.

http://kernel.org/doc/makehelp.txt

If you type `make help`, kbuild emits a page of documentation, and my export scripts put that on the web too.

5 Indexing kernel documentation on the internet

With the kernel's existing internal documentation exported to the web, the next task was adding documentation from the net. Mining the internet for Linux kernel documentation and trying to put it in some coherent order is a huge undertaking, and I barely scratched the surface. What I did find was overwhelming, and had some common characteristics.

There are lots of existing indexes of documentation. Linux Weekly News has an index of all the kernel articles it has published over the years (at `http://lwn.net/Kernel/Index/`). Linux Journal magazine has online archives going back to its first issue (`http://www.linuxjournal.com/magazine`). The free online Linux Device Drivers book has an index (`http://lwn.net/Kernel/LDD3/`). Kernel Traffic has an index (`http://kerneltraffic.org/kernel-traffic/archives.html`). My own mirror of the OLS papers has an index (`http://kernel.org/doc/ols`).

The common theme of these indexes (and many more like them) is that they index only their own local content, because the aim of most of these repositories is to create new local documentation rather than index existing external documents. These indexes are valuable, but collating them together is a nontrivial task. When indexed by topic they don't necessarily use the same topic names, while other indexes are only by date. And this glosses over any actual overlap in the article contents.

Some indexes (such as Documentation/kernel-docs.txt) do link to a number of external sources. Others (such as the Linux Documentation Project `http://tldp.org`) attempt to organize existing documentation by mirroring it. These are valuable resources, but most tend to give up after a certain point, either finding natural boundaries or realizing the enormity of the task of indexing the entire internet and deciding against it. Once promising indexing efforts, such as the Open Source Writer's Group

(at www.oswg.org), the Linux Kernel Documentation Project (`http://www.nongnu.org/lkdp/`), and on "The Linux Kernel: The Book" (`http://kernelbook.sourceforge.net/`), stalled and died.

The Linux Documentation project (`http://tldp.org`) is the largest and most well known of the existing documentation collection projects, but its primary focus is userspace and its method is mirroring. Its efforts go to collecting and mirroring as many documents as possible, not into cross-referencing or deep linking into them.

6 To mirror or not to mirror

The decision whether or not to mirror web resources is tricky, and has no good answer. On the one hand, mirrors get out of synch with their original sources, take up potentially gigabytes of storage, dilute the Google page rank of the original source, raise licensing concerns, often have an inferior user interface to an original page with a style sheet, and so on. On the other hand, resources that aren't mirrored can go 404 and vanish from the net.

The wayback machine at archive.org aims to preserve the entire internet for posterity, and for the most part I chose to rely on that project to preserve information rather than mirroring it. Some things, such as the OLS papers, I chose to mirror in order to present them in a different format (or at a different granularity) than the source material, or because (like the 2006 OLS slides) the sources were already decaying after a relatively short period of time. But where original sources were established and stable, I linked directly to them. (Mirroring the Linux Weekly News archives would be silly.)

7 On old material and editorial ignorance

Deciding whether or not a reference is obsolete requires domain expertise. This is something that an editor often won't have much of, nor time to acquire it. This is another facet of the author vs editor dichotomy.

An editor must accept that there is material they don't understand, and that attempting to become an expert on every topic is not a viable strategy. New content is generated faster than any human being can absorb it without specializing.

As an editor I found myself fielding documentation that I did not have more than the most superficial understanding of. It was not possible for me to improve this documentation, tell if it's up to date, evaluate its accuracy or thoroughness. (In extreme cases, such as foriegn translations, I couldn't even *read* it.)

What can an editor do about this? Pile it up in a heap. Summarize the topic as best they can (which may just be a title statement cribbed from somewhere), link to the documentation they found in whatever order seems appropriate (if all else fails, there's always alphabetical), and wait for people who *do* understand it to complain. If the editor's brief summary plus pile of links does attract relevant domain experts: delegate to them.

In cases where I could tell that a reference was obsolete (such as devfs), I often wanted to link to it anyway. Why? Because it provides historical insight into the current design. "Here's how the kernel used to do things, and here's what was wrong with that approach." A relevant domain expert can avoid reinventing the wheel if they see what was learned from the previous approaches.

So noting the date of older resources in the index can be valuable, but excluding them just because they're old isn't. These days everyone takes for granted that Linux uses ELF executables, but they have to read old articles from 1995 (such as `http://www.linuxjournal.com/article/1139`) to learn why. (Or at least they did until recently, now there's `http://people.redhat.com/drepper/dsohowto.pdf`. Which provides better coverage of the topic? Since I haven't piled up enough links on that topic to worry about pruning them yet, I don't currently have to make that decision.)

8 What's out there?

Here's a quick survey of some of the more prominent kernel documentation sources out on the web. This is not an attempt to be exhaustive, the internet is too big for that.

linux-kernel mailing list archives

The Linux kernel mailing list is the main channel of discussion for Linux kernel developers. The early history of Linux started on usenet's comp.os.minix

and moved to its own "linux-activists" mailing list (archived at http://www.kclug.org/old_archives/linux-activists/ and with a few interesting early posts summarized at http://landley.net/history/mirror/linux/1991.html and http://landley.net/history/mirror/linux/1992.html). It then moved to linux-kernel@vger.rutgers.edu, and eventually to vger.kernel.org when the rutgers machine died.

Numerous archives of linux-kernel are available. The most well known is probably the one at http://www.uwsg.iu.edu/hypermail/linux/kernel/. But the sheer volume of this mailing list is such that few if any kernel developers actually read all of it, and the archives are overwhelming. Drinking from this firehose yields a poor signal to noise ratio; it is a valuable source of raw data, but extensive filtering and summarizing is required to extract useful documentation from it.

The linux-kernel mailing list is one of of almost a hundred mailing lists hosted on vger.kernel.org (see http://vger.kernel.org/vger-lists.html), and many other kernel-relevant mailing lists live on other servers. Although linux-kernel@vger.kernel.org is the big one, others provide plenty of relevant information.

During the course of the documentation fellowship, I posted regular status updates to linux-doc@vger.kernel.org. That list was mostly moribund, so the archives at http://marc.info/?l=linux-doc provide a relatively concise summary of my activities during the project.

http://kernel-traffic.org/

To understand the magnitude of the kernel documentation slush pile, ponder the fate of kerneltraffic.org. This popular, widely-read website provided weekly summaries of discussions on the linux kernel mailing list from January 2000 to November 2005. But eventually the volume overwhelmed editor Zack Brown, who brought the project to an end:

From http://kerneltraffic.org/kernel-traffic/latest.html

Kernel Traffic has become more and more difficult over the years. From an average of 5 megs of email per week in 1999, the Linux kernel mailing list has gradually increased its traffic to 13 megs per week in 2005. Condensing that into 50 or 100 K of summaries each week has started to take more time than I have to give.

Kernel Traffic was an extremely valuable resource due to the sheer volume of material it condensed, and its loss is still strongly felt. These days, most kernel developers consider it impossible for anyone to read all messages on linux-kernel, certainly not on a regular basis.

In 2007 I hired a research assistant named Mark Miller, in hopes of bringing Kernel Traffic up to date. He reproduced the existing site from its XML source files (see http://mirell.org/kernel-traffic), and experimentally summarized a few more weeks. The result was that summarizing each week of posts took him longer than a week, and the amount of expertise necessary to select and summarize interesting threads, plus the sheer number of hours required, made doing just this and nothing else a full time job for someone (such as Zach Brown) who is already a domain expert. The job was simply too big.

Linux Weekly News kernel page

The other systematic summarizer of the linux-kernel mailing list, and the only one to continue a regular publication schedule to this day, is the Linux Weekly News kernel page. Each week, Jonathan Corbet does excellent in-depth analysis of several kernel topics discussed on the list. Since 1999, this has resulted in several hundred individual articles.

The LWN Kernel Index page (http://lwn.net/Kernel/Index/) collects the individual articles and organizes them by topic: Race Conditions, Memory Management, Networking, and so on. Individual articles are linked from mulitple topics, as appropriate.

A second index, the LWN Kernel Page (http://lwn.net/Kernel/), links to article series such as Kernel Summit coverage, 2.6 API changes, and Ulrich Drepper's series on memory management.

The Linux Weekly News kernel page is published regularly, but it does not attempt to be as thorough as Kernel Traffic was. Kernel Traffic provided brief summaries of up to two dozen mailing list threads each week, while LWN Kernel coverage provides in depth articles about 3-5 topics in a given week. The two complemented each other well, and one is not a substitute for the other.

http://kerneltrap.org/

The Kernel Trap website cherry picks occasional interesting threads from the Linux Kernel Mailing List (among other sources) and reproduces them for interested readers. It falls somewhere between Linux Weekly News and Kernel Traffic in content, with an update frequency avereaging less than one article per day.

Kernel Trap takes little time to follow, and the material it highlights is consistently interesting. Some articles, such as "Decoding Oops" (`http://kerneltrap.org/Linux/Decoding_Oops`) every would-be kernel developer should read. But in terms of complete coverage of the Linux Kernel Mailing List, Kernel Trap is the next step down after Kernel Traffic and Linux Weekly News.

In addition to summarizing, KernelTrap also generates new content such as event coverage and interviews with prominent developers (see `http://kerneltrap.org/features`).

Ottawa Linux Symposium proceedings (`http://kernel.org/doc/ols`)

This conference has produced a wealth of high quality kernel documentation over the years. (So much so that despite many hours devoted to absorbing this material, I've personally never managed to read even half of it.)

Due to the high quality of the OLS papers, and my personal familiarity with them, I devoted significant effort to them. After confirming they were freely redistributable, I took the published PDF volumes and broke them up into individual papers (using the `make/splitols.py` script, which uses pdftk). This resulted in over 300 individual PDF files, mirrored on kernel.org.

The next step was to index them. For each year, I created an index file, such as the one at `http://kernel.org/doc/ols/2002`. For the first 25 papers of 2002, I read each one and wrote a brief summary of the contents of each paper, but this was surprisingly exhausting. After that, I just listed the title and authors of each paper. The collective index of the OLS papers links to audio and video recordings of panels, as well as the presentation slides for 2006 (mirrored locally on kernel.org due to their original index pointing into a number of 404 errors after less than a year).

The 2001 papers are no longer available from the OLS website,[1] but Linux Weekly news had copies mirrored. For 2000 I found audio recordings, but no papers. I discovered no material from 1999.

Other conferences

OLS used to be one of four main Linux technical conferences. The others were east and west coast versions of LinuxWorld Expo, and Atlanta Linux Showcase. I have the complete set of casette tapes of talks at the 2001 LinuxWorld Expo which I need to secure the rights to digitize and put online. The CD-ROM from ALS 1999 is at `http://kernel.org/doc/als1999`.

These days, the most important kernel conferences (and the only ones the project's maintainer still regularly attends) are linux.conf.au and the kernel summit. But interesting material can be found at The April 2008 Linux Foundation Summit in Austin, the Consumer Electronic Linux Forum's annual conference in Mountain View, and many others in Europe, Asia, Africa, Japan.... (See `http://www.linuxjournal.com/xstatic/community/events`, `http://elinux.org/Events`, and `http://www.linuxcalendar.com` for lists.)

These other conferences produce lots of material. Videos and presentation files from the 2007 O'Reilly Open Source Convention are at `http://conferences.oreillynet.com/pub/w/58/presentations.html`. Videos from the Embedded Linux Conference Europe 2007 are linked from `http://lwn.net/Articles/266169/`. The full text of 28 papers presented at the Ninth Real-time Linux Workshop (held in Linz, Austria November 2-3 2007) are up at `http://linuxdevices.com/articles/AT4991083271.html`. Things like the Linux Plumber's Conference (`http://linuxplumbersconf.org`) or Japan Regional Technical Jamboree #20 `http://celinuxforum.org/node/82` produce more all the time.

Recently, Usenix released decades of existing material when it opened web access to its conference proceedings. The announcement is at `http://blogs.usenix.org/2008/03/12/`

[1] `http://www.linuxsymposium.org`

This doesn't even get into papers and talks presented at regional LUGs.

Man pages

The best existing documentation on the kernel's system calls is section 2 of the man-pages package. (There's some debate on whether man-pages should document the API exported by the kernel or the wrapped versions provided by glibc, but they're mostly the same.)

I was using Eric Raymond's Doclifter project to convert each new release of man-pages to docbook, and from there to HTML, but eventually the man-pages maintainer started doing his own HTML conversions, and put them on the web at `http://kernel.org/doc/man-pages/online_pages.html`.

This may fall under the heading of "don't ask questions, post errors". Michael Kerrisk had meant to put up html versions of the man pages for some time, but my doclifter conversions were probably horrible enough to bump it up on his todo list before too many outside sources linked to them. (I also sent ESR a few patches to doclifter, and converted his old RCS repository to mercurial so it could go up on the web. This got him started on the whole "source control" kick. Did I mention this project spawns endless tangents?)

Developer blogs and web pages

Prominent kernel developers often have web pages. Many of them (such as `valhenson.com`, `selenic.com/linux-tiny`, and `http://people.netfilter.org/rusty/unreliable-guides/` are documentation resources in their own right.

Many kernel developers also blog. The blog aggregator Kernel Planet `http://kernelplanet.org` does a reasonable job of collecting many developer blogs into a single page, where lots of excellent posts documenting obscure subjects float by... and scroll off the end again, unrecorded.

My own blog for 2007 `http://landley.net/notes-2007.html` documents a lot of my own struggle with the kernel documentation issue. (If you can dig those comments out from the noise about cats and food.)

Wikis and Wikipedia

Distributed, user generated content naturally scales in volume with the size of the userbase. Unfortunately, the editorial task of coordinating, filtering, and integrating the resulting material does not. Doing that takes work.

Drew Curtis, of the news aggregator Fark, recently spoke about "the wisdom of crowds" in an interview:

> We're the only news aggregator out there which is edited, which I think is the next step in social networks because right now everybody is talking about the wisdom of crowds, and all that--which is complete horse ****, and I think the next step is realizing that what crowds pick is pretty much pornography and Internet spam, and as a result you've got to have some editing involved there somewhere.

(Curtis went on to note that the record-holding top story of "Digg" had been puppies playing.)

Wikis are a perfect example of this. The Linux Kernel is the subject of dozens of wikis. Some random examples (with varying degrees of kernel focus) include rt.wiki.kernel.org, elinux.org, linux-mm.org, kernelnewbies.org, unix-kernel-wiki.wikidot.com/linux-kernel-wiki, linux-ntfs.org, wiki.linuxquestions.org, gentoo-wiki.com, wiki.ubuntu.com/KernelTeam, fedoraproject.org/wiki, slackwiki.org, wiki.debian.org, and so on.

The lack of integration leads to multiple wikis emerging even for individual Linux distributions. For example, SuSE has en.opensuse.org, susewiki.org, wiki.linuxquestions.org/wiki/SuSE, suseroot.com, wikipedia's pages on SuSE, the SuSE pages on wiki.kollab.org, linux.ittoolbox.com, www.linuxformat.co.uk/wiki...

The biggest wiki of all is wikipedia, which has hundreds of pages on topics related to the Linux kernel. Unfortunately, the way to find stuff in Wikipedia is to use Google. Wikipedia has both a reluctance to link to anything other than itself, and a general lack of indexing.

Wikipedia's Linux index `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Linux/index` is mostly oriented towards userspace, containing

a single link to the http://en.wikipedia.org/wiki/Linux_Kernel page. That page does not attempt to index pages on numerous kernel-relevant topics (such as Red Black Trees or IPSec).

The related Wikibooks project has several pages devoted to the Linux kernel, of which http://en.wikibooks.org/wiki/The_Linux_Kernel functions as a reasonable index of external resources. It's actually quite nice, and a good source of further links for anyone interested in finding such. However, at the time of writing this paper, it contains exactly three links to wikipedia articles.

None of these wikis really focuses on indexing external content. Few have complete or well-organized indexes even of their own content. They throw data up in the air and leave sorting it up to Google.

Google Tech Talks

I made a small page linking to a few interesting Google Tech Talks (http://kernel.org/doc/video.html) but didn't manage to index even 1% of what's there. And that's just a single video series from one source in California, not a serious attempt to trawl Youtube for kernel content.

I broke down and wrote some

What can I say, I'm weak?

The most popular one hosted on kernel.org/doc would probably be the the Git Bisect HOWTO at http://kernel.org/doc/local/git-quick.html which provides just enough background for somebody to use git bisect without forcing them to learn about things like branches first.

Others (such as Documentation/make/headers_install.txt) went upstream.

(Don't ask about sysfs. It's a heisenberg system: attempting to document it changes the next release. Documenting it by examining its implementation is explicitly forbidden by its developers; you're supposed to read their minds. You think I'm joking...)

Online books

Linux Device Drivers (http://lwn.net/Kernel/LDD3/) is a complete education in the Linux kernel by itself. (I've read maybe 3 chapters.) Linux Kernel in a Nutshell (http://www.kroah.com/lkn/) is also online. "Linux Kernel 2.4 Internals" (http://www.tldp.org/LDP/lki/) remains an excellent if somewhat dated introduction, and the Linux Documentation Project has its own book called "The Linux Kernel" (http://tldp.org/LDP/tlk/tlk.html) based on the 2.0 source. faqs.org has The Linuxx Kernel HOWTO (http://www.faqs.org/docs/Linux-HOWTO/Kernel-HOWTO.html)

I got permission from Mel Gorman to mirror the published version of his book "Understanding the Linux Virtual Memory Manager" at http://kernel.org/doc/gorman. (It's an excellent resource, and I haven't made it past the introduction yet. It's on my todo list.)

Finding new stuff

Jonathan Corbet tracks changes to the Linux kernel on a Linux Weekly News page http://lwn.net/Articles/2.6-kernel-api/ and in the Linux Weather Forecast http://www.linux-foundation.org/en/Linux_Weather_Forecast. Kernel newbies has its own pages http://kernelnewbies.org/LinuxChanges and http://kernelnewbies.org/Linux26Changes). The elinux.org website has its own version of the Linux Weather Forecast (http://elinux.org/Technology_Watch_List).

Many other websites (such as http://www.linuxdevices.com) track changes to their own areas of interest.

And so on

The kernel git archive has some very informative commit messages (browsable at http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git) going back to 2.6.12-rc2. Thomas Gleixner converted the old bitkeeper repository (covering 2.5.0 through 2.6.12-rc2) into git (browsable at http://git.kernel.org/

?p=linux/kernel/git/tglx/history.git). That's just as much an unsummarized firehose as the linux-kernel mailing list, and I never got around to even trying to deal with it.

On http://kernel.org/doc are links to free online copies of the Single Unix Specification version 3, the C99 standard, the ELF and DWARF specs, and more. Go there. Read it.

9 Organizing this heap

If I sound tired just listing all those resources, imagine what it's like trying to collate them.

My approach was to create a large topic index, using nested html "span" tags and a simple python script to create an index from that, and check the lot into a mercurial repository (http://landley.net/hg/kdocs). I could have used wiki software, but kernel.org dislikes active content for security reasons.

Another reason to avoid wiki software is that the public face of the index is HTML, thus the source format (http://kernel.org/doc/master.idx) should be as close to pure HTML as possible. I expected to receive and merge patches against the generated HTML, and those patches needed to apply to the source with as little work on my part as possible.

Once I had a decent topic index (based on examinations of Linux Device Drivers, Linux Kernel Internals, and so on), the next step was to go through the individual subindexes (such as the ones for Linux Weekly News kernel articles, Documentation/, htmldocs, kernel traffic, the Ottawa Linux Symposium papers, and so on) and slot in links to each resource as appropriate. Many resources needed a link from more than one topic, so this was an interative and extremely time consuming process.

I also attempted to summarize each topic briefly, in addition to providing a stack of links. The line between "writing new documentation" and indexing existing documentation was never more blurry than when doing that.

When prioritizing, I kept in mind the rate of churn. Every kernel release breaks out of tree drivers, to the point that even widely used patches such as squashfs or kgdb, applied by every major kernel vendor, are a pain to use with a current vanilla kernel. Documenting interfaces with an expected lifespan of 3 months is a Red Queen's race.

The interface between the kernel and userspace is the most stable part of the kernel, and has some of the best existing documentation. The easiest approach is to start there and work in.

10 Unfinished business

The file http://kernel.org/doc/pending/todoc.txt was my working todo list when the Linux Foundation decided to discontinue the documentation fellowship. After six months, they admitted they hadn't had a clear idea what "solving" the kernel documentation problem meant, and they were going to pull back and reconsider their options. They praised the work I'd done and gave me one more month to finish it up, but did not wish to continue funding it for the full year.

After seven months of drinking from the firehose, I was actually kind of happy to stop. As with the maintainer of Kernel Traffic: it was fun, but I was tired.