Reprinted from the

Proceedings of the Linux Symposium

July 23th–26th, 2003
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
Stephanie Donovan, *Linux Symposium*
C. Craig Ross, *Linux Symposium*

Review Committee

Alan Cox, *Red Hat, Inc.*
Andi Kleen, *SuSE, GmbH*
Matthew Wilcox, *Hewlett-Packard*
Gerrit Huizenga, *IBM*
Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*
Martin K. Petersen, *Wild Open Source, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*

Kernel Janitors: State of the Project

Arnaldo Carvalho de Melo
Conectiva S.A.
acme@conectiva.com.br
http://advogato.org/person/acme

Abstract

The Kernel Janitors Project has been cleaning up the kernel for quite some time, in this paper I'll present what has been done, tasks that kernel hackers added to TODO list, tools used to help in the process, 2.5 changes that need to be propagated thru the tree and other aspects of kernel janitoring.

1 What is the Kernel Janitor Project?

The Kernel Janitors Project grew out of our search for things to help in the development of the Linux kernel, and learning from other patches submitted by more experienced people, we saw that some of these patches indicated error patterns that could exist in other parts of the kernel, we looked and... yes, we discovered that some parts of the kernel suffered from the same problems and in the process we found code bitrotting... I (acme) started maintaining a TODO list for things to fix or clean up and people started submitting suggestions for things to fix that I collected at `http://bazar.conectiva.com.br/~acme/TODO`.

2 A Way for Newbies to Start Hacking the Kernel

Looking at the httpd logs I discovered that lots of people accessed it, so this indeed was something useful as a starting point for people also wanting to help in cleaning up/fixing parts of the kernel.

Several people have the goodwill to help in kernel programming, but many don't have the time to help on areas that require more knowledge and effort, so the Kernel Janitor Project TODO list helps, as there are lots of simple tasks that needs work but are trivial enough to allow those people to help while not requiring much time and effort.

3 The Project Moves to SourceForge

Dave Jones suggested that we moved this to sourceforge, so that we could have it on CVS and allow more people to have admin rights to add more entries, add explanations about the tasks, etc.

I then registered the domain kerneljanitors.org and made it point to the sourceforge web page, that is very simple but has important pointers to resources for new janitors.

4 Project Committers

We're always accepting more people, preferably maintainers of parts of the kernel, so that we can improve the TODO list, and as there is always some boring and repetitive work that the maintainers postpone because there is always more important things to do.

In fact even Linus has posted at least once in the mailing list asking for help on simple repetitive tasks, such as the irqreturn_t conversion for all the interrupt handling routines.

These are the current commiters for this project, at this time:

- `Arnaldo Carvalho de Melo`
- `Dave Jones`
- `Jeff Garzik`
- `Matthew Wilcox`
- `Randy Dunlap`
- `Tariq Shureih`
- `William Lee Irwin III`

5 Mailing list

Using the SourceForge infrastructure we creates the kernel-janitor-discuss mailing list, where we discuss the aspects of the project and review janitor patches.

There has been a continuous arrival of people asking where to start helping, and a number of them became regular janitors.

6 Some Janitors

- `William Stinson`
  - *check_region removal*
- `Art Haas`
  - *C99 struct init style*

7 Tools

There are now several tools that help on pinpointing most of the entries in the TODO list and even some more involved problems that needs hands to fix, I'll briefly talk about several of such tools in the next sections.

7.1 Stanford Checker

Perhaps the first tool for source checking used in the Linux kernel, the Stanford Checker is based on a modified gcc that does several verifications for different types of common problems, and then the Stanford guys do some sanity checking and post the list on the linux-kernel mailing list, where people comment on it, checking if they are false positives and most frequently fixing the bugs.

It is unfortunately an unreleased tool, but it has been of great help over the last years.

7.2 kj.pl

This was a very simple perl script that Dave Jones wrote, that searched for some very trivial problems, but Dave has stopped working on it as now we have smatch.

7.3 smatch

Dan Carpenter's smatch is a released tool that allows developers to write perl scripts to search for problems, it is also based on a modified gcc.

Dan has created the kbugs.org web site where he has several smatch scripts and a list of the

results of those scripts, allowing janitors to pick real problems to work on.

The current scripts, for reference, are:

- `ReleaseRegion`
- `ReleaseIRQ`
- `DoubleSpinlock`
- `Dereference`
- `GFP_DMA`
- `UnreachedCode`
- `SpinlockUndefined`
- `FunctionStack`
- `UncheckedReturn`
- `SpinSleepLazy`
- `UnFree`

I'll quote Dan now just as an example of smatch results in kbugs.org:

"There were quite a few smatch related fixes in the 2.5.69 kernel. Someone fixed 4 SpinSleepLazy bugs, 3 UnreachedCode bugs and 6 unchecked calls of `copy_to_user()`. It's not entirely clear who did what from the changelog, but thank you anonymous heros."

7.4 cqual

Cqual is another tool that can help finding problems on codebases such as the kernel, here is the project description, from its authors:

"Cqual is a type-based analysis tool that provides a lightweight, practical mechanism for specifying and checking properties of C programs. Cqual extends the type system of C with extra user-defined type qualifiers. The programmer adds type qualifier annotations to their program in a few key places, and Cqual performs qualifier inference to check whether the annotations are correct. The analysis results are presented with a user interface that lets the programmer browse the inferred qualifiers and their flow paths."

It is part of a bigger project at the University of Berkeley called Open Source Quality, this seems to be a project that deserves investigation by janitors as it wasn't mentioned up to now on the kjp mainling list.

7.5 sparse

And now to something unreleased at this time: Linus Torvalds's sparse tool:

"Sparse is a semantic parser of source files: it's neither a compiler (although it could be used as a front-end for one), nor is it a preprocessor (although it contains as a part of it a preprocessing phase).

It is meant to be a small—and simple—library. Scanty and meager, and partly because of that easy to use. It has one mission in life: create a semantic parse tree for some arbitrary user for further analysis. It's not a tokenizer, nor is it some generic context-free parser. In fact, context (semantics) is what it's all about—figuring out not just what the grouping of tokens are, but what the *types* are that the grouping implies."

It is indeed very interesting that more and more people are working towards having tools that can help in improving the quality of Open Source projects such as Linux.

8 Documentation

- lwn.net Articles by Jonathan Corbet
- Tariq's "Drivers DOs and DONTs"

- Other Articles for janitors
- Arjan’s article about how not write a driver
- Greg KH’s article.