5: Practical suggestions 1: process interaction

Jonathan Corbet
LWN.net
corbet@lwn.net

What this section is about

An attempt to share some experience on how things go wrong.

Here we'll focus on process issues.

Failures

Looking at failures is instructive

- So we'll do some of that

Please note:

- No disrespect is intended

“A bridge, under its usual conditions of service, behaves simply as a relatively smooth level surface on which vehicles can move. Only when it has been overloaded do we learn the physical properties of the materials from which it is built.”

-- Herbert Simon

“Hey, all my other theories made sense too. They just didn't work. But as Edison said: I didn't fail, I just found three other ways not to fix your bug.”

-- Linus Torvalds

How can one avoid failing?

Let your developers participate

Community-connected developers are:

- Happier
- More productive
- More influential

Attend developer conferences

Linux-Kongress
LinuxCon
FOSDEM
linux.conf.au
FISL
Linux Plumbers Conference
...

You'll learn...

What's happening in the community

Developments of interest to you

...

Who your peers are

Develop skills in house

Chasing established developers is expensive and difficult

It's not a zero-sum game.

Getting started

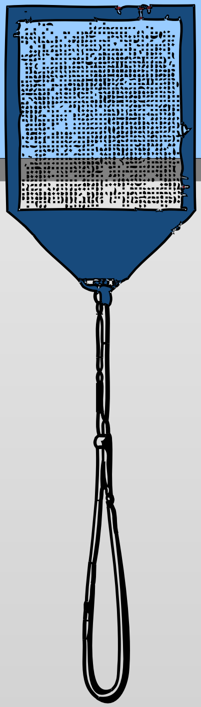
The “kernel janitors project” has a TODO list

- Ignore it

Please do not start posting white space fixes.

Getting started

The #1 project for all kernel beginners should surely be “make sure that the kernel runs perfectly at all times on all machines which you can lay your hands on.”
-- Andrew Morton



In other words:

Fix bugs

Getting started

Review code

You can learn a lot from reading and understanding other people's code. Study the things posted, and ask why things are done specific ways, and point out problems that you have noticed. It's a task that the kernel really needs help with right now.
-- Greg Kroah-Hartman

Design your processes around participation

“We can't release any code which has not been through internal QA” is a recipe for disaster.

Communicate your plans

Early!

Case study: Tux3

A next-generation filesystem by Daniel Phillips

2008-07-23	Initial announcement
2008-11-25	Booting as root filesystem
	...
2009-08-16	Last commit

"Do NOT fall into the trap of adding more and more stuff to an out-of-tree project. It just makes it harder and harder to get it merged. There are many examples of this."

-- Andrew Morton

Daniel kept adding features

...then lost interest

“Anyway, Andrew Morton was right, we should have merged into mainline as soon as Tux3 was booting as root.”

-- Daniel Phillips

Lessons

- Out-of-tree code is nearly invisible
  - Few users
  - Few contributors
  - Little momentum

Photo: Team Traveller

Lessons

Get it into the mainline early!

Seek influence, not control

Case study: em28xx

...a video4linux driver

2005-11-08	Initial driver merge
...	
2008-01-05	Markus Rechberger's final em28xx patch
2008-11-02	Replacement patch rejected
2009-08-09	Markus's final kernel patch

“Companies should be aware that if they try to submit any code to you they will loose the authority over _their_ work.”
-- Markus Rechberger

Another example

May, 2004

- Hans Reiser tries to block the addition of new functionality to reiserfs.

“The fact is, maintainership does _not_ mean ownership. It means that you should be _responsible_ for the code, and you get credit for it, but if problems happen you do NOT “own” it. Not at all.”
  -- Linus Torvalds

Lessons

Contributing means losing control

Others *will* improve your code

Photo: Yuliya Libkina

Seek influence, not control

Influence comes from
- community participation
- code contributions

Dan Frye's advice:
- Have your developers immersed in the community

Case study: the deadline scheduler

Con Kolivas's scheduler rewrite

2007-03-04	First post
2007-03-05	Linus amenable to merging
2007-03-19	Linus gets irritated
2007-04-13	Molnar posts CFS
2007-07-10	CFS merged for 2.6.23
2007-07-25	Con leaves the kernel community

Understand that some things are easy to merge

Drivers!
Obscure architecture-specific code
...

Some things are harder

Ooh you have a VM patch that helps swap on the desktop! I can help you here with my experience from swap prefetch:

1: Get it reviewed and have noone show any evidence it harms
2: Find hundreds of users who can testify it helps
3: Find a way of quantifying it
4: ...
5. Merge into mainline

I haven't figured out what 4 is yet. I believe it may be goto 1.

-- Con Kolivas

Expect delays for...

Memory management changes
Core filesystem work
Security policies
Scheduler changes
...

Improve the kernel for everybody

...or at least don't make it worse

Seek outcomes, not credit

The best solution might not be yours

Dan Frye again:

IBM rewards engineers who push a solution forward regardless of whether their code is merged.

Participate in the wider discussion

-ck list did not help

Mailing lists

Linux-kernel is intimidating
- 500 messages/day
- Variable politeness

But: it's where things happen

Options:
- Filter heavily
- Read LWN

Subsystem lists

Many of them exist

- netdev
- linux-mm
- linux-scsi
- ...

vger.kernel.org/vger-lists.html

Subsystem lists

Can be easier environments

Sometimes the only place to be
- netdev

Not popular with all developers
- Tend to hide conversations and problems
- Can make interactions harder

When in doubt: copy linux-kernel

List etiquette

Never remove Cc's

> Hey, I was reading that! Please do *not* go making modifications to Cc: lists. Just do reply-to-all and be happy, thanks.
> -- Andrew Morton

Copy others liberally

Do not assume they will see something on the list.

No top-posting

Good subject lines

Volume on l-k is huge and the best strategy is to get mail recognized as relevant and to have reviewers' estimate of priority before looking into the thing more or less close to that after. We all have heuristics; defeat these too often and you will *become* one.
-- Al Viro

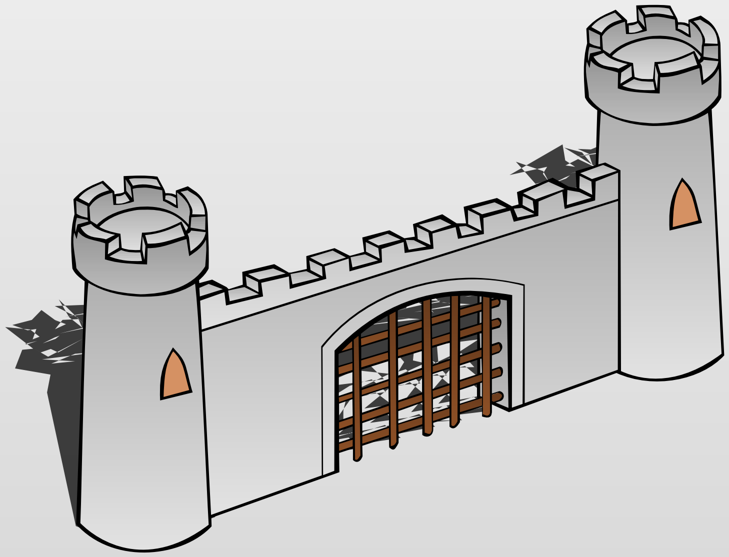
One other thing...

Avoid internal lists!

Have discussions in public whenever possible.

Follow through

“Throwing it over the wall” is not appreciated



...but it's better than nothing.

Don't break things

Case study: 2.5.x IDE

2002-02-15	Martin Dalecki's first “IDE cleanup” patch
2002-03-08	IDE18, subsystem takeover
2002-08-09	IDE115 merged
2002-08-16	Martin quits, all IDE work reverted

“Breakage is the price you have to pay for advancements”

-- Martin Dalecki

Breaking things is a bad idea.

Case study: reiser4

2002-10-29	First code post
2003-07-24	2.6.0-test merge request
2004-08-19	Added to 2.6.8.1-mm2
2005-09-11	Push for 2.6.14
2006-07-20	Push for 2.6.19
2006-10-11	Hans Reiser arrested

What were the problems?

Non-POSIX filesystem behavior
Numerous technical difficulties
Hard-to-reproduce benchmarks
Antagonistic approach to others
Memories of reiser3

Linux is not a research system

Visionary brilliance will not excuse a poor implementation

Lessons

It's better not to accuse others of conspiring against you

Photo: Rob!

Don't take it personally

They don't hate you

...or your company

...or your objectives

The community remembers past actions

Developers also think far into the future

Photo: krupp

Market changes to developers

Case study: SystemTap

2003-11	DTrace debuts
2005-10	RHEL4 introduces SystemTap
2008-07	FTrace merged
2009-06	Perf Events merged
2009-09-22	SystemTap 1.0 released
????	SystemTap merged

2008 Kernel Summit

50% had tried to use SystemTap
20% succeeded

“I thought everyone learned the lesson behind SystemTap's failure: when it comes to tooling/instrumentation we don't want to concentrate on the fancy complex setups and abstract requirements drawn up by CIOs as development isn't being done there. Concentrate on our developers today, and provide no-compromises usability to those who contribute stuff.”

-- Ingo Molnar

In other words...

If kernel developers don't see the value

...it won't go in.

Related case study: TALPA

Posted in August 2008

- Never merged as such

The goal:

- Provide hooks for virus scanners

Problems with TALPA

Kernel developers disliked it
- Why bother with broken security models?

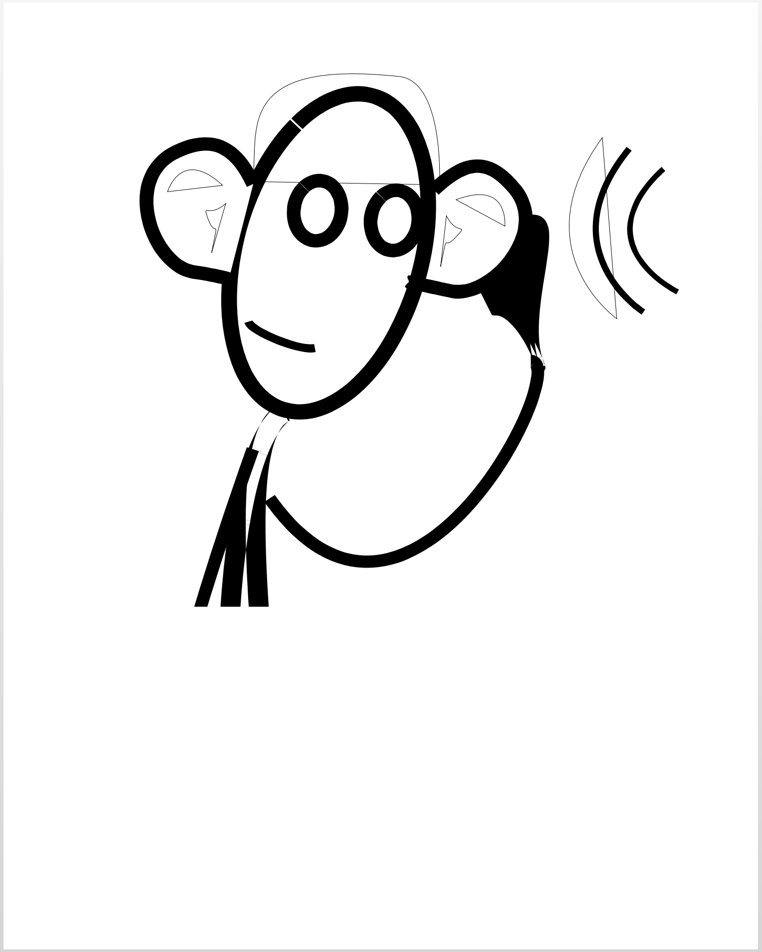
Badly-expressed requirements
- No threat model
- Solutions not needs

Communicate your requirements

What is the problem to be solved?
Use cases?

Requirements should be “what,” not “how.”

Listen



Enter fanotify

Merged in August, 2010 (2.6.36)

Provides hooks for virus scanners

What changed?

Featured a cleanup of file event notification

- Replaced inotify and dnotify

Rephrased requirement:

- “Enable virus scanners to hook into file operations without using rootkit techniques.”

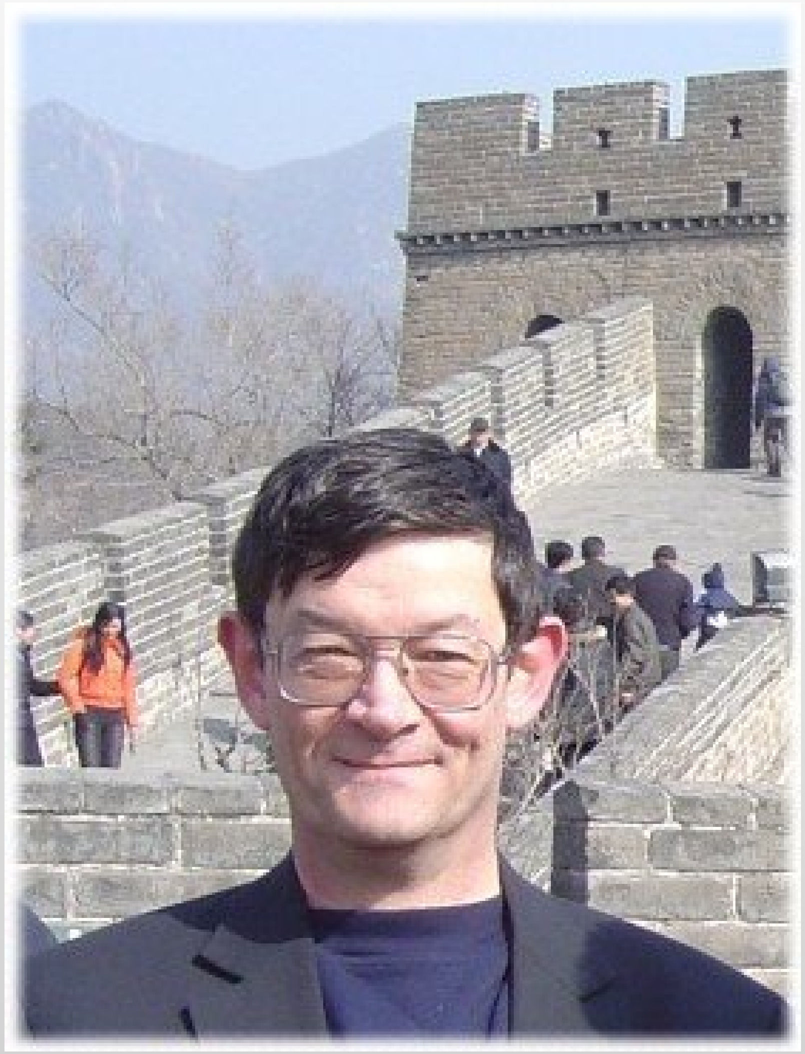
Lessons

Patches must be sold to developers

- Not managers or customers

Cleaning things up builds goodwill

What if things seem blocked?



Andrew Morton

...is the maintainer of last resort.

In conclusion

The process may seem full of hazards

...but it's not that hard

Common sense and listening will see you through

...most of the time

Questions?