Buried alive in patches: 6 months of picking up the pieces of the Linux 2.5 kernel

Dave Jones
SuSE Labs
davej@suse.de, http://www.codemonkey.org.uk

Abstract

When development began on the 2.5 Linux kernel, I volunteered to forward port 2.4 fixes to 2.5, and keep them in sync ready for when Linus was ready to accept them. Additionally, I collected the small bits Linus frequently overlooked, and perhaps more importantly, tried to keep the 2.5-dj tree in a usable, bootable state even when 2.5 mainline wasn't.

1 Introduction

With the advent of a new development series of the Linux kernel, Linus Torvalds typically hands off the current tree to someone else, who becomes maintainer of that stable series, and work on the development branch accelerates, whilst the stable branch collects fixes and updates rather than new features.

Typically, as focus on the development branch is in other areas, important fixes continue to pour into the stable series which are sometimes not picked up in the development branch until much later, or sometimes, not at all.

In the past Alan Cox has done sterling work in picking up the fixes that go into stable series, and collecting them together in his regularly released -ac patches. At various points, Alan would then push known good parts to Linus at such a time that he is ready for them.

When the 2.4 kernel diverged to the 2.5 development branch, there was still a considerable number of fixes going into 2.4. Alan was busy with other projects at the time, and so it was suggested that 'someone' should pick up the bits going into 2.4 and make sure they don't get left behind for 2.5.

Without fully thinking through the consequences, I decided to step forward, and got a lot more than I bargained for, but learned a lot, and had a lot of fun on the way.

2 The problems

The easy part of the job looks something like this:

```
repeat:
$ cd linux-2.5
$ cat ../patch-2.4.18-pre1.diff \
  | patch -p1 -F1 --dry-run

(note rejects)

$ vi patch-2.4.18-pre1.diff

(chop out rejects)
until applies
```

The same process applies whenever Linus releases a new 2.5 kernel.

This is however just a fraction of what the job entails.

The first thing to be aware of is that a lot of the fixes going into 2.4 may not be relevant to what's happening in 2.5. For example, maybe the maintainer of relevant code wants things fixed cleaning in 2.5, whereas a band-aid is acceptable in 2.4, or maybe large restructuring of the code is planned for 2.5, so the fix is irrelevant. Sometimes development of a driver continues actively in 2.4 and 2.5, and takes wildly different directions.

Keeping up-to-date with what every maintainer is doing with their subsystem is a tricky task that involves lots of mindreading, guesswork, and occasionally email to ask, "What exactly is going on with xxx?"

Another tricky part of the job is making sure the tree is still in a state where you can test that what you've just merged actually works. Not easy at times in a development series when there are bits of core functionality being ripped apart. Sometimes this results in compromises (not merging certain parts until they compile), sometimes getting ahead of mainline (where fixes appear faster than Linus merges them), and sometimes by means of adding really ugly hacks that don't stand a chance to be accepted for mainline, but "do the job" for most people who are more concerned about their own specific part of the kernel.

Perhaps by far trickiest part of all however is trying to split things up into Torvalds-size pieces so that every so often, a resync can occur to push some of the more obviously correct, and well-tested bits back to the mainline tree. As I found my feet with syncing, I tried various approaches, some of which worked out better than others.

There are several reasons for the difficulty.

- In the case of merging a 2.4pre to 2.5, using the above method means I'm left with a several MB patch which originally consisted of perhaps dozens of smaller patches. Whilst some of these are sent to the Linux kernel mailing list, not all are visible until they show up in Marcelo's tree.
- Maintainer issues. Sometimes it's not immediately obvious from reading the diff (or even the code in a before and after state) why a patch is needed. The maintainer however knows (or at least should know) his/her own code inside out, and know the precise reasoning behind every diff going into Marcelo/Linus' kernel. In these circumstances, it's often the best policy to let them take care of merging such patches, as they can explain to Linus in much better terms why he needs to take the patch instead of my guesswork and hand waving.
- Patch drift. Patches I did manage to pick up from the kernel mailing list, or were Cc:'d to me were a little easier, as they tended to come with good descriptions by the patch author. The only problem with these was that over time, the patch would no longer apply to Linus' vanilla tree, so the patch would have to be kept up to date. With so many patches applied, keeping them all up to date seperately became harder and harder, especially if several patches wanted to touch the same files.
- Conflicts. When two or more patches are touching the same file, what happens next depends on the level of change in the various parts. For example, if I had several patches touching the tulip network driver, one fixing an obvious bug, one fixing a spelling mistake, and another adding

a MODULE_LICENSE tag, the latter two are trivial enough that the patch doesn't need splitting.

Where there are two parts to the patch fixing different problems, or perhaps adding functionality, things get more complicated, and tools such as editdiff become huge timesavers.

It became apparent very quickly that splitting up the several MB patches from 2.4 back into their component parts for each release wasn't feasible due to the amount of time it took. Each time a new Marcelo prepatch appeared, it was merged wholesale after removal of unneeded parts. When the periodic resyncs with Linus then occured, there were in many cases quite a few trivial patches to the same file, making it easy to get rid of lots of the smaller parts of my tree.

3 Patch Rejection

It would be an easy job to simply apply every patch that ever gets sent either directly, or to the kernel mailing list. However things are never simple, and a number of factors have to be taken into consideration.

Chances of Linus ever accepting them. Some patches are just too ugly to live. Various people sent me patches that Linus had rejected, in the hope that as it was coming from me, Linus would somehow take a different view. Somewhat amused by this, most of these patches are either memorable, or discussion with the relevant maintainer is usually enough to get a "don't apply" message back. If there's no chance of Linus ever taking it, then keeping patches of this kind in my tree was deemed pointless.

Controversial patches. A good example of this case was Eric Raymond's CML2 patch. A very large patch that touched the configuration file of every part of the kernel. It's hard to imagine a more far-reaching change. At one point, Linus even made claims that he had no interest in the kernel configuration language, and that it would be probably better maintained outside the kernel tree. So this example also falls into category 1. Had I merged CML2 at any point, this would have made it impossible to merge any configuration updates from mainline without first rewriting them as CML2 rules, which was unacceptable. Likewise, any changes made could not be sent back to mainline.

Orthoganal works. Sometimes patches appear, and there will be nothing technically wrong with them, but perhaps the timing is wrong due to someone else working in the same area on maybe a larger scale. An example of this was the i386 subarch patches James Bottomley did, which unfortunatly clashed with the considerable rewrites Randy Dunlap did to i386-specific drivers such as MTRR.

4 Timeline of events.

4.1 December 2001

At the beginning of December, Linus had put out 2.5.0, and was concentrating almost solely on merging Jens Axboe's block layer rewrite. At the same time, Marcelo had begun his first 'real' patch merging, after having put out the rush-released 2.4.16. It was noted that these fixes were not getting merged into 2.5, and Dave Miller suggested that someone collect them, and keep them up to date until such a time that Linus was ready to accept them. I had been doing this partially at the time for my own use anyway, so I decided to take it on.

Towards the end of the month, when Linus was up to 2.5.1pre11, I made my first release of -dj, which was around a 1MB diff against Linus' current tree.

4.2 January 2002

In January, Linus had got up to 2.5.2pre, and Marcelo had accelerated in patch merging as he got used to his new role. The diffsize between my tree and Linus' had started to increase dramatically, and so the first resync of the trees was planned. After pushing a considerable amount of the more obvious fixes to Linus, only some of the trickier-to-merge bits remained, although still a sizable amount.

It was at this time that Linus started breaking things dramatically in 2.5. First came the big kdevt redesign, which broke compilation of many parts of the kernel. The linux-kernel mailing list was awash with many fixes for these compile errors, although it took Linus some time to get many of them merged.

Another key point of note in January was the introduction of the new Framebuffer API into my tree. James Simmons wanted a 2.5 that stood more chance of being able to compile, and decided to use my tree as a basis for development.

A great supplement to me picking up various small patches on the Linux kernel mailing list was Rusty Russell's trivial patchbot. Patches sent to the robot get archived, and automatically retransmitted on the sender's behalf, and do all kinds of magic like automatically checking that they still apply when a new kernel appears, and bouncing a "doesn't apply any more—please rediff" back to the patch author. With both Rusty and myself picking up and retransmitting these on the patch-author's behalf, the "Linus isn't taking my patches" arguments for trivial patches all but disappeared.

During pushing bits to Linus, I discovered Tim Waugh's patchutils, which is a set of tools for manipulating diffs. Until this point, when sending a patch to Linus, I had been doing pretty much everything in vim, and chopping out unnecessary parts of the rest of my tree from the patch. Quite a time-consuming process. With patchutils, things got a lot easier as I could now with a 'simple' command line extract all the related parts of a patch from my tree. For example,

```
grepdiff pf_gfp_mask dj1.diff \
        | xargs -n1 \
        filterdiff dj1.diff -i
```

would extract all the diffs from my tree that touch pf_gfp_mask. After doing this, and deleting the irrelevant hunks of the the diff from the output, the patch was more or less ready to go to Linus.

As well as these, Patchutils also includes many other useful tools that save more time to a patch-merger than any other tool I've yet to run.

Toward the end of the month, my tree was around 2MB away from Linus.

4.3 February

At the beginning of February, Linus was up to 2.5.5, and with the block layer taking shape, he had started merging some other large new features. This kernel saw the introduction the x86-64 port, the ALSA merge, and the first parts of the new input layer (all of which had been in -dj for a month).

On the downside, some other things had continued to break dramatically. Lots more drivers no longer compiled due to the virt_to_bus macros being changed in an attempt to get more portable drivers. I made a compromise

in my tree and wrapped this change in an option called CONFIG_DEBUG_OBSOLETE. When not selected, the old behaviour occurred, and the drivers continued to compile. Crude, but effective.

Other notable changes this month included the various IDE cleanups by Martin Dalecki, Vojtech Pavlik, Pavel Machek, and others. As a result of this work, Andre Hedrick stood down as 2.5 IDE maintainer.

It became aparent that I hadn't pushed to Linus for a while, as by now, my tree was 4MB away from Linus, with quite a lot of patches pending. The new framebuffer API work was taking up a large percentage of this, as was the new input layer work.

Christoph Hellwig had been regularly looking through my patches, and with nearly every release he would spot something really dumb that I did, like reintroducing calls to a now-dead API, or CVS $ID: tag damage. (I use CVS and frequently forget to add files with -ko). These silly mistakes happened often enough that I decided to write a simple perl script to check for silly mistakes like this.

```
#!/usr/bin/perl -w
# checkdiff.pl -- 2.5-dj kernel patch checker.
#
# I'm stupid.
# This script sanity checks diffs before I put them out, so
# that hch doesn't have to remind me I goofed.

use strict;

foreach (@ARGV) {
    process ($_) or warn "Couldn't check file $_: $!";
}

sub process {
    my $filename=shift;

    open INPUT, $filename or return undef;
    my @lines=<INPUT>;
    close INPUT;
    chomp @lines;

    my $linenr=0;

    foreach my $line (@lines) {
        $linenr++;

        if ($line=~/davej/ and $line=~/\$Id:/) {
            print "Found davej CVS damage at line $linenr\n$line\n\n";
        }

        # We are adding a line, check its not obsolete.
        if ($line=~/^\+/) {
            if ($line=~/iorequest_lock/) {
                print "Adding code to frob iorequest_lock" .
                      " at line $linenr\n$line\n\n";
            }
            if ($line=~/[        ]MAJOR[ (]/) {
                print "Adding code to frob MAJOR at line $linenr\n$line\n\n";
            }
            if ($line=~/get_fast_time/) {
                print "Adding get_fast_time at line $linenr\n$line\n\n";
            }
            if ($line=~/strtok/) {
                print "Adding strtok at line $linenr\n$line\n\n";
            }
            # ... other rules here ...
        }
    }
    return 1;
}
```

Cut-down version of the Perl script used for checking patches before uploading to kernel.org

4.4 March

By March, mainline was up to 2.5.7. Probably the biggest change was the introduction of Robert Olsen and co's NAPI work.

In an attempt to get the diff size between my tree and Linus' down, I decided to drop the arch updates for architectures like S390, m68k, etc. The reasoning behind this was that even with the updates from my tree, they never compiled in 2.5 anyway, due to the lack of other necessary changes, such as those imposed by the introduction of Ingo Molnar's O(1) scheduler.

Linus had at this point been using bitkeeper for a few weeks, and was just starting to get comfortable with it. Patches seemed to be getting applied at a much more accelerated rate than previously. I wondered if it could help out with the merging of my-tree to Linus', and played with it for a week or so, often exchanging ideas/queries with Larry McVoy, but ultimately, it didn't work out for me. The only way bitkeeper would work for such a large set of diffs (at the time, up to 7MB away from mainline) would have been to have many trees for all the different patches, all combined into one union tree. There was however a problem with this. If I go to the extreme of splitting up my tree into component parts to feed into bitkeeper trees, I may as well just feed those small bits to Linus as regular GNU patches.

Linus agreed with this, and this is how we continued the merging.

I started to fall behind a little in this month with syncing both from my tree to Linus and vice versa. Mostly due to me moving house, and having to make do without life's essentials (like ADSL) for a while.

After getting back online, and spending a few days getting everything back up to date, Linus went on holiday for a few weeks, with the parting note "Jeff Garzik and Dave Jones will be taking care of patches whilst I'm gone." The following two weeks were mostly quiet fortunately, which gave me great opportunity to split up patches ready for the largest resync so far.

4.5 April

April began with 2.5.8 still being current. Linus had returned from his holiday, and 'The big resyncing' ensued. With 7.5MB of diff between his tree and mine, this was no small task, and not one that would be over any time soon.

I pushed 128 patches to Linus, 493KB worth, and a further 474KB in 50 patches to Jeff Garzik (network drivers and the like). 6 hours later, 126 of the patches I sent direct to Linus were applied. An hour later, most of what I sent to Jeff also showed up in Linus' tree. In a state of shock, I wondered how I could "shovel faster." By the time Linus put out pre1, my tree was 6MB away from mainline.

pre2 was another few hundred KB (although mostly fixing wrong merges from pre1). Whilst splitting up various bits for Linus, and looking through the patches in my tree, I also found a lot of other patches that really didn't make sense to continue carrying (whitespace differences, superseded patches, updated CVS idents, etc.).

A mistake that happened during merging here was that CONFIG_DEBUG_OBSOLETE slipped in as a Config.help text. Interestingly, Linus decided to drop what this option was wrapping in my tree, and make it unconditional. Whilst having more portable drivers was an admirable goal, it hadn't really prompted any of the maintainers to fix their drivers, causing more headache than anything productive.

After this, I spent a week at Linuxworld—i.e.,

where I had hoped to do more patch splitting, but it didn't happen due to time constraints. What did happen was the beginning of more catch up work. By the time I returned, I had to resync 2.4.19pre4,pre5,pre6 & 2.5.8pre2 & pre3 to my tree.

In addition to this, there were now 113 patches in my inbound queue. At this point it was realised that I couldn't 'stop to do a resync' every so often, and that syncing a "moving target" was considered only viable option.

Toward the end of the month, Linus stopped doing -pre patches, and instead started doing full releases more often. At the end of the month, 2.5.11 was current, which featured another large number of merges, including the beginnings of the new framebuffer code. At time of writing this paper, resync against 2.5.11 hadn't been done, but estimated diffsize was between 4-5MB.

5 Future Plans

At the time of writing, there are over two months remaining before OLS, during which much more resyncing will happen, and conversely, many more patches are likely to get applied to my tree. The larger parts of my tree (the Framebuffer code, new input API, ALSA OSS updates) are slowly starting to get merged into mainline. If by some freak opportunity my tree manages to get down to a reasonable size again, I'll take a look at using bitkeeper for merges once again.

Given a hypothetical perfect world, the plan is that by the time 2.6 is ready, the -dj series of kernel patches will become less necessary. Depending on rates of merging, positions of moon, and other acts of randomness, it may be that 2.6-dj patches will become necessary until fully merged, and some of the more experimental bits may end up being put back until 2.7.x opens.

6 Links

`http://www.kernel.org/pub/people/davej/patches/2.5/`
Where my -dj patches are to be found.

`http://www.codemonkey.org.uk/Linux-2.5.html`
A list of known remaining problems with current/recent 2.5's.

`http://diary.codemonkey.org.uk/`
My online diary where I frequently write about progress on 2.5

`http://cyberelk.net/tim/patchutils/`
Tim Waugh's patchutils

`trivial@rustcorp.com.au`
Rusty Russell's trivial patch robot.

7 Acknowledgments

The work of patch integrator is largely a solitary task, but there are countless people worthy of gratitude, from the many people who were prepared to download and try out my -dj patches, especially those who sent feedback/reports/patches.

Tim Waugh for patchutils, which made merging so much easier.

Larry McVoy for making Linus scale despite the critics.

And finally, Alan Cox for being busy with other things in December 2001. Without which, I'd probably have found something much more boring to have spent the last 6 months hacking.

Proceedings of the
Ottawa Linux Symposium

June 26th–29th, 2002
Ottawa, Ontario
Canada

Conference Organizers

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

Proceedings Formatting Team

John W. Lockhart, *Wild Open Source, Inc.*