

# Re: [PATCH] Open Firmware device tree virtual filesystem

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-12/msg08104.html>

---

- *From:* Mitch Bradley <[wmb@xxxxxxxxxxxxxx](mailto:wmb@xxxxxxxxxxxxxx)>
  - *Date:* Sat, 30 Dec 2006 23:36:45 -1000
- 

David Miller wrote:

...  
Can we please not have N different interfaces to the open-firmware calls so that perhaps powerpc and Sparc have a chance of using this code too?

The base interface function is `callofw()`, which is effectively identical to `call_prom_ret()` in `arch/powerpc/kernel/prom_init.c`. So it seems that PowerPC could use it. I suppose I could change the name of `callofw()` to `call_prom_ret()`, thus making the base interface identical to PowerPC's. All it does is argument marshalling, translating between C `varargs` argument lists and the OFW `argarray` format.

SPARC should be able to use that same base interface function directly. It is written to the standard OFW client interface. The x86 client interface that I tested it on is essentially the same code that is in OBP. It wouldn't work on ancient Sun machines with the `sunmon romvec` interface, but Sun stopped making such machines something like 16 years ago.

On sparc and powerpc, we even build an in-kernel data structure of the entire open-firmware device tree that code like your's could use if you make a simple setup layer for i386 as well. We have interfaces for modifying property values at run time too.

I would strongly suggest looking at things like `arch/{sparc,sparc64,powerpc}/kernel/prom.c` and `include/asm-{sparc,sparc64,powerpc}/prom.h` and `arch/{sparc,sparc64,powerpc}/kernel/of_device.c` and `include/asm-{sparc,sparc64,powerpc}/of_device.h` since we've already invested a lot of thought and infrastructure into providing interfaces to this information on powerpc and the two sparc platforms.

I did look at those files, until my eyes glazed over. In powerpc land, the files that are the underlayer for `proc_devtree.c` comprise 4700 lines of code (the files you list plus `prom_init.c`). In sparc land, it is only 3200 lines (the files you list plus the prom interface library). On top of that, `proc_devtree.c` is 233 lines.

## Re: [PATCH] Open Firmware device tree virtual filesystem

In contrast, ofw\_fs.c is 261 lines, and the base interface function callofw() is 97 lines (half of them comments).

Admittedly, this is something of an apples-to-oranges comparison, because ofw\_fs only exports a read-only device tree and nothing else. But in the case where that is all you need, a direct interface to OFW that avoids the middleman seems like a good choice.

I did consider first creating a memory data structure identical to the powerpc/sparc one, but that looked like it was going to be essentially twice as much code for no extra capability. The code to traverse the device tree and create the memory data structure is roughly the same as the code to create the filesystem structure. I just didn't see the value of an intermediate representation for systems that don't otherwise need it. (A setup layer would have let me use proc\_devtree.c directly, so the total amount of new code would have been the same, but many people told me that if I even suggested using procs the kernel gurus would blow me out of the water without bothering to blink.)

In the SPARC and PowerPC spaces, Open Firmware is widespread, so it makes sense for those kernels to use OFW extensively. In x86 land, OFW is far from being the dominant firmware, so the x86 kernel is unlikely to depend on OFW services at a deep level. That being the case, the deep-integration features of the sparc and powerpc OFW interfaces are not needed in x86 land. But a lightweight interface to the device tree is certainly useful for the platforms that do have OFW. It might be useful for other processors as well, especially on platforms that don't need the deep configurability that drove the OFW design.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>