

Re: Generic battery interface

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-08/msg00012.html>

- *From:* "Shem Multinymous" <multinymous@xxxxxxxxxx>
 - *Date:* Tue, 1 Aug 2006 01:34:30 +0300
-

Hi Jean,

On 8/1/06, Jean Delvare <khali@xxxxxxxxxxxxx> wrote:

Disclaimer: I have no idea how the input interface works currently. And I don't know what problem you are trying to solve. I just thought my hwmon-oriented comments might help.

Your comments are highly relevant! Almost everything said in this thread, apart from "where should we put battery readouts", is applicable to hwmon.

The problem we're trying to solve is minimizing system load and timer interrupts caused by apps that track kernel-issued readouts (e.g., hwmon's), without the application making unnecessary assumptions about the driver or vice versa.

- > 1. A new ioctl DELAYED_UPDATE, with parameters min_wait and min_fresh, meaning: "I want an a fresh readout. If I poll() this FD
- > with POLLIN then send an input-ready event at time is T+min_wait, or
- > when you have a readout that was received from the hardware at time
- > T+min_fresh, whichever is *later*. Likewise if I select()".
- > Here T is the time of the ioctl call and min_wait>=min_fresh.

"A or B, whichever is later", effectively means "A and B". Or am I missing something?

The equivalent phrasing using "and" is s follows:

"as soon as the current time is at least T+min_wait *and* you have a readout that was received from the hardware at time at least T+min_fresh."

Re: Generic battery interface

I fail to see the difference between `min_wait` and `min_fresh`.

Here's an example illustrating all parameters.

Suppose the app does a `DELAYED_UPDATE` with `min_wait=700` and `min_fresh=1000` at time `T`, and then `poll()`s with a timeout of `9999`.

At time `T+600` the driver receives update events (yeah, it's that kind of driver). If this is all that happens, the app will remain blocked for `9999ms` — the requirement for a readout fresher than `T+700` is never fulfilled.

If the driver gets a second event at time `T+800`, the app will be unblocked at time `T+1000`. If, instead, the driver gets the second event at time `T+1200`, the app will be unblocked immediately.

The `hwmon` interface (`sysfs`s now, `procfs`s before) has been returning cached values by default for years. Changing this at this point might be confusing.

Yes. All those should be told to use `O_NONBLOCK`, or be delayed by one readout cycle whenever they poll an attribute. How serious is this?

The essential problem is that we don't want drivers to query the hardware and cause timer interrupts when nobody's listening, so cached values can get arbitrarily stale. Thus, an out-of-the-blue read by a **must** wait for a refresh (i.e., hardware query). I don't see a non-kludgy way out of it.

I don't see much benefit in waiting for updated values compared to reading them from the cache. The driver knows better how frequently it can read from the chip.

Yes, but the driver doesn't know how frequently apps **need** it to read from the chip.

Take the `hdaps` driver, for example. The hardware can provide fresh readouts at a rate of `500Hz`. Some apps (e.g., `hdapsd`) actually need a high poll rate. Others (e.g., joystick emulation and `hdaps-gl`) work nicely with `20Hz`. And the `tp-theft` app will be useful even with `1Hz`, and may change its poll rate depending on circumstances. So, what is the driver supposed to do?

In my proposal, the driver and all app (implicitly) negotiate a

Re: Generic battery interface

poll rate which makes sense for all parties involved.

And no hardware monitoring chip I know of can tell when the monitored value has changed – you have to read the chip registers to know.

Yes, this may not be relevant for traditional hwmon chips, but we're trying to handle event-based data sources too. You might get an ACPI event on temperature change, or a "critical temperature" interrupt (which flips a boolean sysfs attr), etc. I can't think of a really convincing example, but we certainly want an interface that will support such drivers transparently to userspace.

> To illustrate, here's an example of a proper polling loop (sans error
> checking). This app wants to refresh its display when the data has
> changed, but not more often than once per second. It wants the
> readouts to be reasonably spaced: they should be obtained at least 700ms
> apart. And it needs to update its GUI every 3 seconds regardless of
> readouts.

I don't see the point in the 700ms rule. If you don't want new data more often than once per second, the readouts will be spaced by one second, which implies > 700ms already.

Only on average. If you use min_wait=1000 and min_fresh=0, you might be seeing cached readouts that were obtained at times
999,1001,2999,3001,4999,5001,...
which is probably not what you want.

No seek(fd, ..., 0) here? sysfs files are supposed to be simple text files, aren't they?

Yes, for sysfs there's a seek(fd, ..., 0). There isn't one if you use the input infrastructure's approach, where read()s return a stream of event records.

Shem

–

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/>

Re: Generic battery interface