

Re: Possible bug in keyboard.c (2.6.10)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-01/9100.html>

From: Roman Zippel (zippel_at_linux-m68k.org)

Date: 01/30/05

Date: Sun, 30 Jan 2005 21:13:56 +0100 (CET)

To: Vojtech Pavlik <vojtech@suse.cz>

Hi,

On Sat, 29 Jan 2005, Vojtech Pavlik wrote:

> > *That doesn't necessarily mean we have to pack everything into a single
> > structure. E.g. we present pci boards with multiple functions as multiple
> > devices, the same can be done for input devices. More important is that
> > kernel drivers only expect a certain class of devices, e.g. the keyboard
> > driver only needs the keyboard related parts and would also allow us to
> > add more keyboard specific callbacks instead of sending events.*
>
> *The USB HID devices often are crossing the device type boundaries. A
> keyboard with a few mouse buttons, for example. Yes, we could split it,
> but I really doubt the gain.*

Why?

> *The GGI people went the way of predefined device types ...*

And what is this supposed to tell me?

> > *This is rather a hint that the abstraction is wrong, e.g. why not send
> > sound events to the `pckbd_handler`? A device should just send input
> > events and handlers handle them and with sysfs we should actually rename
> > the latter to `input_drivers` (it's less confusing this way).*
>
> *You imply that the `atkbd` module would register itself both as an input
> device, and as an input handler? That's quite an interesting idea I
> haven't thought about before.*

That's one idea I'm playing with, but the keyboard.c needs a serious cleanup first.

> *But then all the handlers also have to register themselves as devices,
> for generating the LED and Sound events. Probably we then could get rid
> of the handlers altogether and have devices only, which would both send
> and receive events.*

Linux-Kernel: Re: Possible bug in keyboard.c (2.6.10)

>

> *It would work, but to me this looks even more confusing.*

No, you have to clearly define the path of events, if you want to a two way communication via input events, you would need two devcies and two drivers, but I doubt we really want that.

E.g. led handling should be no events at all, they are actually properties of the keyboard device and if the keyboard driver wants to set them, it should just do `dev->setleds(...)`. I know of at least two keyboards (an Amiga and a Mac keyboard), where the caps status (and therefore the caps led) cannot be changed via software.

> > *I have a patch which makes the keymap data more*

> > *dynamic and assigns it to keyboard device,*

>

> *Cool! How do you load the keymaps for the specific devices? How do you*

> *assign the devices to vcs?*

The loading part is still missing. Right now I only concentrate on killing the global data in keyboard.c

> *Can a user have per-vc keymaps?*

Why would you want this?

> > *which has the positive side effect that all keyboards don't have to*

> > *send the same keycodes anymore (and we don't have to break all the*

> > *keyboards anymore which don't use AT style keycodes).*

>

> *Well, I think the fact that the input layer uses an unified set of*

> *numbers for the keys really helps in a lot of places. One is that you*

> *don't need (architectures * languages) of keymaps and only can define*

> *language keymaps.*

For new devices that maybe okay, but that doesn't mean we have to break old devices, which were working just fine.

A small tool that converts from mapping to another is probably easier and it doesn't completely removes the need for arch specific keymaps, e.g. my Amiga has no NumLock key, so I use a different shift key two switch to the alternative keypad mapping. Assuming all the world is a PC never really worked, if you want to write portable code.

> *However, I'm sure that X won't be happy to receive anything but the*

> *single rawmode it expects.*

>

> *A question for you: What is raw mode good for? (I'd like the emulation*

> *to go, and not be replaced by anything at all.)*

I don't mind fixing X, but why not make raw events available if they exists (if only for debugging), if they have to be emulated, you can still create an emulation device, which can be feed them back to the keyboard

driver.

>>>> *– fine grained matching/filtering: I have no idea why the input layer has
>>>> to do this down to the single event instead of just the event type.
>>>
>>> I wonder what do you mean by this, the layer itself doesn't have any
>>> codepaths dependent directly on event code, just on the types.
>>
>> E.g. also `input_match_device`.
>
> That's there for the handlers to be able to decide whether a device is
> interesting for them. Since we don't have device type information (and
> USB HID devices will necessarily not provide us with it, although often
> they do), we can't simply assign handlers based on the device type.*

Why can't you do this in the connect method? Most drivers should just care about the device type and take all events of a certain type and pass it on to userspace somehow.

> *The setup of the bitfields is there `_primarily_` for userspace.*

So why not export it via sysfs?

> *Then the userspace will need to know what the device is, which features
> it has, and how to handle it. We have mice, touchpads, touchpoints,
> touchscreens, keyboards, tablets, joysticks, ... and each need special
> care. Some touchpads have palm detection, other don't. Some mice don't
> have three buttons, the middle button emulation is desired in that case.*

I don't mind device properties, but do we have to do this in such great detail? A lot of simple devices don't have this information, why force them to make this information up? Not all mouse drivers actually know whether the mouse has a third mouse button. If a device has extra information, then simply export it via sysfs, but it's not needed at runtime.

Kernel drivers only care about whether a device generates key/mouse/joystick/... events, and this doesn't require detailed bitfields, e.g. in the case of a keyboard the keybit array is just duplicated information from keycode array (which means extra code just to keep them in sync).

bye, Roman

–

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>