[PATCH 0/11] LED Class, Triggers and Drivers

Source: http://linux.derkeiler.com/Mailing−Lists/Kernel/2006−01/msg11423.html

- *From*: Richard Purdie <rpurdie@xxxxxxxxx>
- *Date*: Tue, 31 Jan 2006 13:41:22 +0000

This is an updated version of the LED class/subsystem I proposed a while ago. It takes John Lenz's work and extends and alters it to give what I think should be a fairly universal LED implementation.

Hopefully a decision on whether this is going to head into mainline or not can be made soon. I've not had any feedback from Russell on this issue (but have asked).

I'm unsure what the kconfig name for the class should be (something better than NEW_LEDS is needed but arm is using LEDS).

The series consists of several logical units:

* LED Core + Class implementation
* LED Trigger Core implementation
* LED timer trigger (example of a complex trigger)
* LED device drivers for corgi, spitz and tosa Zaurus models
* LED device driver for locomo LEDs
* LED device driver for ARM ixp4xx LEDs
* Zaurus charging LED trigger
* IDE disk activity LED trigger
* NAND MTD activity LED trigger

Changes
=======

The patches have been updated after the feedback from GregKH and others on LKML and other feedback elsewhere. These include:

* using class_create()/class_destroy()/class_device_create()
* updates to the timer trigger so it accepts a duty parameter
* addition of led drivers for tosa and ixp4xx
* using an enum for led brightness
* correct the ide trigger when ide_disk isn't compiled
* added some brief documentation

Why?

====

LEDs are really simple devices usually amounting to a GPIO that can be turned on and off so why do we need all this code? On handheld or embedded devices they're an important part of an often limited user interface. Both users and developers want to be able to control and configure what the LED does and the number of different things they'd potentially want the LED to show is large.

A subsystem is needed to try and provide all this different functionality in an architecture independent, simple but complete, generic and scalable manner.

The alternative is for everyone to implement just what they need hidden away in different corners of the kernel source tree and to provide an inconsistent interface to userspace.

Other Implementations
=====================

I'm aware of the existing arm led implementation. Currently the new subsystem and the arm code can coexist quite happily. Its up to the arm community to decide whether this new interface is acceptable to them. As far as I can see, the new interface can do everything the existing arm implementation can with the advantage that the new code is architecture independent and much more generic configurable and scalable.

I'm prepared to make the conversion to the LED subsystem (or assist with it) if appropriate.

Implementation Details
======================

I've stripped a lot of code out of John's original LED class. Colours were removed as LED colour is now part of the device name. Multiple colours are to be handled as multiple led devices. This means you get full control over each colour. I also removed the LED hardware timer code as the generic timer isn't going to add much overhead and is just as useful. I also decided to have the LED core track the current LED status (to ease suspend/resume handling) removing the need for brightness_get implementations in the LED drivers.

An underlying design philosophy is simplicity. The aim is to keep a small amount of code giving as much functionality as possible.

The major new idea is the led "trigger". A trigger is a source of led events. Triggers can either be simple or complex. A simple trigger isn't configurable and is designed to slot into existing subsystems with minimal additional code. Examples are the ide−disk, nand−disk and

zaurus−charging triggers. With leds disabled, the code optimises away.
Examples are nand−disk and ide−disk.

Complex triggers whilst available to all LEDs have LED specific
parameters and work on a per LED basis. The timer trigger is an example.

You can change triggers in a similar manner to the way an IO scheduler
is chosen (via /sys/class/leds/somedevice/trigger).

So far there are only a handful of examples but it should easy to add
further LED triggers without too much interference into other
subsystems.

Known Issues
============

The LED Trigger core cannot be a module as the simple trigger functions
would cause nightmare dependency issues. I see this as a minor issue
compared to the benefits the simple trigger functionality brings. The
rest of the LED subsystem can be modular.

Some leds can be programmed to flash in hardware. As this isn't a
generic LED device property, I think this should be exported as a device
specific sysfs attribute rather than part of the class if this
functionality is required (eg. to keep the led flashing whilst the
device is suspended).

Future Development
==================

At the moment, a trigger can't be created specifically for a single LED.
There are a number of cases where a trigger might only be mappable to a
particular LED. The addition of triggers provided by the LED driver
should cover this option and be possible to add without breaking the
current interface.

A cpu activity trigger similar to that found in the arm led
implementation should be trivial to add.

Richard

–

- ***Follow−Ups***:
  - ***Re: [PATCH 0/11] LED Class, Triggers and Drivers***
    - ◊ *From:* Pierre Ossman
  - ***Re: [PATCH 0/11] LED Class, Triggers and Drivers***
    - ◊ *From:* Jan−Benedict Glaw

- Prev by Date: ***Re: [PATCH] prevent nested panic from soft lockup detection***
- Next by Date: ***[PATCH 6/11] LED: Add LED device support for the zaurus corgi and spitz models***
- Previous by thread: ***[PATCH 3/11] LED: Add LED Trigger Support***
- Next by thread: ***Re: [PATCH 0/11] LED Class, Triggers and Drivers***
- Index(es):
  - ***Date***
  - ***Thread***