What: /sys/firmware/acpi/interrupts/
Date: February 2008
Contact: Len Brown <lenb@kernel.org>
Description:

All ACPI interrupts are handled via a single IRQ,
the System Control Interrupt (SCI), which appears
as "acpi" in /proc/interrupts.

However, one of the main functions of ACPI is to make
the platform understand random hardware without
special driver support. So while the SCI handles a few
well known (fixed feature) interrupts sources, such
as the power button, it can also handle a variable
number of a "General Purpose Events" (GPE).

A GPE vectors to a specified handler in AML, which
can do a anything the BIOS writer wants from
OS context. GPE 0x12, for example, would vector
to a level or edge handler called _L12 or _E12.
The handler may do its business and return.
Or the handler may send send a Notify event
to a Linux device driver registered on an ACPI device,
such as a battery, or a processor.

To figure out where all the SCI's are coming from,
/sys/firmware/acpi/interrupts contains a file listing
every possible source, and the count of how many
times it has triggered.

```
$ cd /sys/firmware/acpi/interrupts
$ grep . *
error:       0
ff_gbl_lock:       0   enable
ff_pmtimer:       0  invalid
ff_pwr_btn:       0   enable
ff_rt_clk:       2  disable
ff_slp_btn:       0  invalid
gpe00:       0  invalid
gpe01:       0   enable
gpe02:     108   enable
gpe03:       0  invalid
gpe04:       0  invalid
gpe05:       0  invalid
gpe06:       0   enable
gpe07:       0   enable
gpe08:       0  invalid
gpe09:       0  invalid
gpe0A:       0  invalid
gpe0B:       0  invalid
gpe0C:       0  invalid
gpe0D:       0  invalid
gpe0E:       0  invalid
gpe0F:       0  invalid
gpe10:       0  invalid
gpe11:       0  invalid
gpe12:       0  invalid
```

```
gpe13:       0  invalid
gpe14:       0  invalid
gpe15:       0  invalid
gpe16:       0  invalid
gpe17:    1084   enable
gpe18:       0   enable
gpe19:       0  invalid
gpe1A:       0  invalid
gpe1B:       0  invalid
gpe1C:       0  invalid
gpe1D:       0  invalid
gpe1E:       0  invalid
gpe1F:       0  invalid
gpe_all:    1192
sci:    1194
sci_not:     0
```

sci - The number of times the ACPI SCI
has been called and claimed an interrupt.

sci_not - The number of times the ACPI SCI
has been called and NOT claimed an interrupt.

gpe_all - count of SCI caused by GPEs.

gpeXX - count for individual GPE source

ff_gbl_lock - Global Lock

ff_pmtimer - PM Timer

ff_pwr_btn - Power Button

ff_rt_clk - Real Time Clock

ff_slp_btn - Sleep Button

error - an interrupt that can't be accounted for above.

invalid: it's either a GPE or a Fixed Event that
doesn't have an event handler.

disable: the GPE/Fixed Event is valid but disabled.

enable: the GPE/Fixed Event is valid and enabled.

Root has permission to clear any of these counters. Eg.
echo 0 > gpe11

All counters can be cleared by clearing the total "sci":
echo 0 > sci

None of these counters has an effect on the function
of the system, they are simply statistics.

Besides this, user can also write specific strings to these

files
to enable/disable/clear ACPI interrupts in user space, which can
be
used to debug some ACPI interrupt storm issues.

Note that only writting to VALID GPE/Fixed Event is allowed,
i.e. user can only change the status of runtime GPE and
Fixed Event with event handler installed.

Let's take power button fixed event for example, please kill
acpid
and other user space applications so that the machine won't
shutdown
when pressing the power button.

```
# cat ff_pwr_btn
0       enabled
# press the power button for 3 times;
# cat ff_pwr_btn
3       enabled
# echo disable > ff_pwr_btn
# cat ff_pwr_btn
3       disabled
# press the power button for 3 times;
# cat ff_pwr_btn
3       disabled
# echo enable > ff_pwr_btn
# cat ff_pwr_btn
4       enabled
/*
 * this is because the status bit is set even if the enable bit
is cleared,
 * and it triggers an ACPI fixed event when the enable bit is
set again
 */
# press the power button for 3 times;
# cat ff_pwr_btn
7       enabled
# echo disable > ff_pwr_btn
# press the power button for 3 times;
# echo clear > ff_pwr_btn       /* clear the status bit */
# echo disable > ff_pwr_btn
# cat ff_pwr_btn
7       enabled
```