

Re: [RFC/PATCH 0/22] W1: sysfs, lifetime and other fixes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-04/6348.html>

From: Evgeniy Polyakov (johnpol_at_2ka.mipt.ru)

Date: 04/26/05

To: dtor_core@ameritech.net

Date: Tue, 26 Apr 2005 11:19:16 +0400

On Mon, 2005-04-25 at 16:32 -0500, Dmitry Torokhov wrote:

> On 4/25/05, Evgeniy Polyakov <johnpol@2ka.mipt.ru> wrote:

>>

>> *How do you suppose to notify about alarm condition?*

>> *Not from bus layer?*

>> *Who does send "link is down" messages? It is not the same*

>> *as device is present and found, it like "w1 device has something to read".*

>> *For example w1 ds18s20 thermal sensor may send information*

>> *about "85 degree problem" - it is read when sensor did not*

>> *finished temperature transformation yet, how non-bus layer may know about it?*

>

> *Classes. And return -EAGAIN when reading is not ready (for some reason).*

Device may be broken, btw, and return that value always.

How to notify about such condition?

>>

>> *Your idea about classes over the various buses is good,*

>> *but unfortunately it is utopia, at least for now,*

>

> *It is not an utopia, it is Linux kernel and we do not need to get in*

> *unfinished solution.*

That will require too many changes to say:

"hey, tomorrow we will have new generic w1/i2c/superio set".

I believe it can be done, it is definitely a good idea,

but let's fix existing bugs before changing the whole tree.

>> *so let's create w1 core layer (which, btw, is not only bus,*

>> *which is managed by bus master driver, but also some logic over it,*

>> *one may call it w1 stack, stack can send such a messages, doesn't it)?*

>>

>

> *Heh ;) Let's concentrate on sensors stack, please...*

If we still have a bugs there, I do not think it is good idea to move things ahead.

I repeat, I agree that unified class hierarchy for sensors devices is a very good idea, but it can not be implemented in a couple of days, so I would postpone it until others bugs are fixed.

> > > >

> > > > *As I said I have feature requests for ability to export w1 devices outside w1 core.*

> > > > *Probably it is due to it's private non-GPL usage, so it is not created, but it is usefull feature actually and we can not know what will happen in what context when we export master/slave devices.*

> > >

> > > *Look at your present w1_remove_master_device. It sleeps. Sop there is no need for a separate thread, callers must be able to sleep anyway.*

> >

> > *That is exactly why control thread exists – to manage sleepable operations!*

>

> *Read what I wrote again – if caller is sleeping on thread's completion there is really no need for a thread. The caller can do whatever thread does. Only if caller would send request and continue one would need to set up a thread.*

Why do you think caller is in process context?

Control thread was created to process that work, which can not be done in interrupt context, i.e. some command from unknown context is deferred to control thread execution, which process it in a safe context. Using connector for exaple you may send command REMOVE_*, which will be received in BH context and just set the flag, which will be seen by control thread.

> > > > *w1 slaves can be found on the bus without search method reaction*

> > > > *implemented in it's asic, btw.*

> > > > *And it is _very_ usefull to add/remove slaves using external command but not using*

> > > > *automatic detection in search methods.*

> > >

> > > *But the request for that will come from userspace with is perfectly*

> > > *able to sleep. You are over-engineering and making kernel code*

> > > *unnecessarily complex without thinking it through.*

> >

> > *Connector's requests come from BH context.*

> > *Only module unloading comes with good context (in our case we do not get read/write operations),*

> > *but I do not want to limit the system only for that kinds of events.*

> >

>

> *And you still have master's thread to manage slaves. Although I don't quite understand why you would need manual addition of slaves. Either they speak W1 protocol and will be added automatically, or they don't speak w1 and then they are not w1 devices.*

There are w1 devices that do not respond to search command
[actually there are devices that understand only one command at all],
ok, they are broken, but it does not matter.

>>>>> *I have feature requests for both adding/removing and exporting
>>>>> master devices and slaves to the external world.*
>>>>>
>>>>> *External as in userspace? It (user thread) can wait just fine...*
>>>>>
>>>>> *Exporting them into other kernel modules.*
>>>>> *We do not know in what context that structures will be used there.*
>>>>>
>>> *Why other kernel modules would be interested in raw access w1_slaves?*
>>> *C are to give an example?*
>>
>> *Concider w1 battery slave device, which exports unified interface to
>> the userspace.*
>> *It requires access that can be obtained from different generic module
>> [like existing kernelspace/userspace protocol,
>> proper device files and so on],
>> which will implement only read/write interface.*
>> *It's read method will get_slave_by_something() and read it's data
>> or do something else, then generic module will use that data.*
>>
>> *Generic butterfly monitor will not scan w1 bus for it's devices,
>> since it even does not know about w1, it only understands reading/writing
>> operations.*
>>
>
> *And here you are thinking of classes again. Writing separate battery
> monitor applets for i2c, w1, superio and the rest is not less silly.
> You are trying to move them over your ocnconnector code. Alternatively
> you could have move them to class codes and build netlink notification
> on top of them. This way you'd separate buses (physical interface)
> with userspace interfaces and allowed use of different transports.*

No, connector is just an example.

Classes hierarchy for all sensor devices is good idea,
but not too easy to be implemented.
I strongly agree with it and would like to port w1 to it.

Connector could be used to control that classes – just an example.

>>>
>>> *Wrong level. You need to start with device implementing w1_bus_master
>>> (w1_bus_ops) to remove dangling data structures). Easiest way I think
>>> it have the driver compiled as a module and remove it altogether – why
>>> keep it if you don't need master?*
>>>
>>> *1. master can be removed by command. It is not in process' context.*

> > *Current thread can remove only all object at once, but nevertheless*
> > *I do not want to limit it.*
>
> *You are also need to remove code controlling physical device presented*
> *as w1_master. The request will go do a different system (module).*

No need to remove bus master device, it can be there with zero refcnt,
so when it will call w1_remove_master_device() it wouldn't block.

> > *2. control thread can add/remove new slaves by request.*
> > *Usefullness of that ability was pointed in my previous e-mail,*
> > *but you skipped that part.*
>
> *I am still missing usefullness of it.*

There are devices that can not be found using w1 search commands,
there are devices that only understand one command,
I agree they are broken, but they still can be easily supported
by w1 subsystem.

> > >
> > > *The less code in kernel that produces data availavle elsewhere the better :)*
> >
> > *Does 3 lines of code for reading slave's names is too big*
> > *price for not scanning the whole /sys/bus/w1/w1_master1/ directory?*
> >
>
> *How often do you use them? While debugging only?*

Always.

With long line it is very common to lose w1 slaves – that is why it was
TTL
attribute created – it's purpose is to integrate such events.
With device like iButton it _IS_ thy _only_ behaviour – slave object
exists only couple of seconds.

>
> > > > > > *w1-master-cleanup.patch*
> > > > > > *Clean-up master device implementation:*
> > > > > > *– get rid of separate refcount, rely on driver model to enforce*
> > > > > > *lifetime rules;*
> > > > > > *– use atomic to generate unique master IDs;*
> > > > > > *– drop unused fields.*
> > > > >
> > > > > *That patch is very broken.*
> > > > > *I completely against it:*
> > > > > *1. it breaks process logic – searching can be interrupted and stopped,*
> > > > > *thread will exit on signals.*
> > > > >
> > > > > *Interrupted/stopped from userspace?*
> > > > >

>>> *Your loop waits only until interrupt happens – it can be delivered from
>>> anywhere.*
>>>
>>> *No, only root can kill kernel thread so it is pretty safe. And hey, if
>>> a thread goes mad maybe it's a good thing that it can be killed.*
>>>
>> *And if it exits – it breaks the logic – user can not know the state
>> of the master device when thread is exited, but module was not removed
>> by request.*
>>
>>
> *I (as a root) have zillion ways to break the system. There is not hing
> new. The oint that an ordinary user can't do anything with that
> thread.*

Signal can be sent not only by your request.

>>>
>>> *device model is here to _use_ it. It already implements bunch of stuff
>>> you have to re-implement if you do it "your own way" and it is already
>>> debugged much better than your solution. And if there is a problem
>>> with driver core implementation – well, more people are looking at it
>>> and are more likely to discover a problem and offer a solution.*
>>>
>>> *I do not understand why you are against full integration with device
>>> model – it does simplifies and unifies the code.*
>>>
>> *Because it is not needed here – and even if it could be integrated
>> more closer – your changes broke too many special design cases,
>> which are not acceptable.*
>>
>>
> *Having too many special design cases in otherwise pretty simple bus
> indicates that there something wrong with the design.*

I see your point.

>>>> *With such changes how to increment slave's usage counter? module_get
>>>> (w1_family)?*
>>>>
>>>> *Actually if you need it it would be get_device(&w1_slave->dev). And if
>>>> you need pin family object you would get
>>>> get_driver(&w1_family->driver). But I don't think you will needed it.
>>>> Actually, at some point I had w1_family_get() implemented as a wrapper
>>>> to get_driver() but since it does not seem to be needed I dropped it.*
>>>>
>>>> *We can not do it in that way.*
>>>> *1. low-level bus master code differs from what w1_master is.*
>>>> *2. family itself is different from slave object.*
>>>>
>>>> *Having that we need to create w1_master/w1_family device/driver model*

Linux-Kernel: Re: [RFC/PATCH 0/22] W1: sysfs, lifetime and other fixes

> > *locking + w1_bus_master/w1_slave locking or move them to device/driver*
> > *model too. Why it is needed I still do not see, there is*
> > *proper locking schema, which is broken in the places where*
> > *existing model touches device/driver one, and it will be fixed.*
> >
>
> *You have exactly the same problem with master devices too. What*
> *escapes me is the desire to have 2 separate refcounting for the same*
> *object. Except for ability to introduce 2x more bugs.*

Yep.

As I said in previous e-mail – I do want to remove objects in any time, so it has it's own locking schema, but I also do want to use sysfs objects, so there is another locking schema – in the place they are touch each other there is a problem and I will fix it, probably by moving object freeing into →remove() callback.

--

Evgeniy Polyakov
Crash is better than data corruption -- Arthur Grabowski

–

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/>

- application/pgp-signature attachment: This is a digitally signed message part