

Re: newbus integration of MOD_QUIESCE

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-07/0045.html>

From: Robert Watson (rwatson_at_freebsd.org)

Date: 07/14/04

Date: Tue, 13 Jul 2004 20:22:43 -0400 (EDT)

To: "M. Warner Losh" <imp@bsdimp.com>

On Tue, 13 Jul 2004, M. Warner Losh wrote:

> : *MOD_STRONGUNLOAD* – *Unload even though you're in use. Detach the driver,*
> : *deadfs the file system, wither the geom, sever the sockets, etc. May*
> : *cause disruption, but may also veto, depending on the subsystem,*
> : *especially if the subsytem has no way to notify its consumers of*
> : *impending doom. Can be vetoed, but try harder before vetoing. Some*
> : *subsystems might always return EBUSY for this if there's really no way*
> : *to express "undesirable departure" upwards.*
>
> *This is tue current MOD_UNLOAD*

Well, sort of. It might be well true of device drivers, but in the world of synthetic network interfaces, etc, the MOD_WEAKUNLOAD semantics apply.

> : *MOD_QUIESCE* – *Attempt MOD_WEAKUNLOAD, and if that fails, ask the module to*
> : *start draining in some form. I'm a bit unclear on quite what's*
> : *intended, but this seems to be less atomic notion than "unload, or*
> : *don't" at various points on the spectrum. I.e., it kicks off a state*
> : *transition in what is likely a slightly poorly defined state machine.*
> : *Right now, the state machine is "Not loaded", "Loaded", and we use a*
> : *lock to prevent intermediate states from colliding.*
>
> *This is the heart of my questions about MOD_QUIESCE.*

For some module systems this is free, or at least designed in. In the MAC Framework, an unload request is treated as a blocking quiesce request. A module can block unload if it wants to, but by default, the MAC Framework will wait (possibly a long time) for the framework to get to a point where it can safely unload. I.e., it assumes unload can be made atomic, if you wait long enough.

With others, that's clearly not the case. You identified a couple of interesting edge cases, such as "State transition driver into 'shutting down', now do something that requires the driver", but I think there's some other big problems with the state machine. What if an application

freebsd-arch: Re: newbus integration of MOD_QUIESCE

needs the functionality — does it ask the driver to change its mind, wait for the driver to unload so it can load it again, or can the unload and load overlap? How long might it have to wait? "Drain, oh, unless I need it" may mean one thing for a driver, but something else for a file system. The nice thing about the current model is that it supports two positions: on, and off.

This raises another edge case — the case today where we load a driver, there's a linking or initialization error, and we leave it loaded in a broken state. It becomes more apparent when you load a driver with a mutex initialization in it's load routine. If you've never given it a try, try loading if_tap.ko into a kernel with the tap interface compiled in.

Robert N M Watson FreeBSD Core Team, TrustedBSD Projects
robert@fledge.watson.org Principal Research Scientist, McAfee Research

freebsd-arch@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"