

Re: Forcefully unmounting devfs...

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-12/0095.html>

From: Kirk McKusick (mckusick_at_mckusick.com)

Date: 12/21/04

To: "Poul-Henning Kamp" <phk@phk.freebsd.dk>

Date: Tue, 21 Dec 2004 12:29:00 -0800

> To: Robert Watson <rwatson@freebsd.org>
> From: "Poul-Henning Kamp" <phk@phk.freebsd.dk>
> In-Reply-To: Your message of "Tue, 21 Dec 2004 18:23:46 GMT."
> Date: Tue, 21 Dec 2004 19:35:00 +0100
> Message-ID: <81690.1103654100@critter.freebsd.dk>
> Cc: arch@freebsd.org
> Subject: Re: Forcefully unmounting devfs...
> X-ASK-Info: Whitelist match
>
> In message <Pine.NEB.3.96L.1041221182235.48875Y-100000@fledge.watson.org>, Robe
> rt Watson writes:
>>
>> On Sat, 18 Dec 2004, Poul-Henning Kamp wrote:
>>>
>>> Brainteaser for the x-mas days:
>>>
>>> What is the correct handling of busy device vnodes belonging
>>> to a devfs mountpoint which is being forcefully unmounted ?
>>>
>>> If you think this has a simple answer, please don't bother replying.
>>
>> Being something of a traditionalist, I'd ask the question: what happens in
>> 4.x when you forceably unmount a UFS file system out from under the device
>> nodes? I'm guessing we deadfs the UFS vnodes, which results in varying
>> degrees of havoc depending on the device type?
>>
>> It's worse: we specfs the vnodes which results in varying degress of success,
>> depending on device type.
>>
>> --
>> Poul-Henning Kamp | UNIX since Zilog Zeus 3.20
>> phk@FreeBSD.ORG | TCP/IP since RFC 956
>> FreeBSD committer | BSD since 4.3-tahoe
>> Never attribute to malice what can adequately be explained by incompetence.
>>
>> _____
>> freebsd-arch@freebsd.org mailing list

freebsd-arch: Re: Forcefully unmounting devfs...

- > <http://lists.freebsd.org/mailman/listinfo/freebsd-arch>
- > *To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"*

Poul-Henning is correct. To elaborate, the operations vector for device vnodes have historically been built up from a mix of specfs operations which deal with the mechanics of doing I/O (read, write, strategy, ioctl, etc) and the containing filesystem (UFS, NFS) operations for naming (open, stat, chown, chmod, rename, etc). When the containing filesystem is forcibly unmounted, the naming operations are stripped away leaving only the I/O operations. Thus read, write, strategy, and such continue to work, but name related operations on the descriptor (fstat, fchown, fchmod, etc) will fail as the underlying naming operations are gone. I still believe that this is a reasonable approach as it lets things like the disk continue to operate when an unmount is done.

The code to detect device aliases was there in part to ensure that if a new device node showed up, it would be associated with an existing opened device vnode rather than resulting in the creation of a new vnode. The effect was to reassociate the orphaned vnode from the previously mounted filesystem with the new naming information.

Kirk McKusick

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"