
Subject: Re: [RFC][PATCH 3/4]: Enable multiple mounts of /dev/pts

Posted by [Pavel Emelianov](#) on Fri, 15 Feb 2008 07:57:38 GMT

[View Forum Message](#) <> [Reply to Message](#)

sukadev@us.ibm.com wrote:

> Pavel Emelianov [xemul@openvz.org] wrote:

> | Serge E. Hallyn wrote:

> | > Quoting Pavel Emelyanov (xemul@openvz.org):

> | >> [snip]

> | >>

> | >>>> Mmm. I wanted to send one small objection to Cedric's patches with mqns,

> | >>>> but the thread was abandoned by the time I decided to do-it-right-now.

> | >>>>

> | >>>> So I can put it here: forcing the CLONE_NEWNS is not very good, since

> | >>>> this makes impossible to push a bind mount inside a new namespace, which

> | >>>> may operate in some chroot environment. But this ability is heavily

> | >>> Which direction do you want to go? I'm wondering whether mounts

> | >>> propagation can address it.

> | >> Hardly. AFAIS there's no way to let the chroot-ed tasks see parts of

> | >> vfs tree, that left behind them after chroot, unless they are in the

> | >> same mntns as you, and you bind mount this parts to their tree. No?

> | >

> | > Well no, but I suspect I'm just not understanding what you want to do.

> | > But if the chroot is under /jail1, and you've done, say,

> | >

> | > mkdir -p /share/pts

> | > mkdir -p /jail1/share

> | > mount --bind /share /share

> | > mount --make-shared /share

> | > mount --bind /share /jail1/share

> | > mount --make-slave /jail1/share

> | >

> | > before the chroot-ed tasks were cloned with CLONE_NEWNS, then when you

> | > do

> | >

> | > mount --bind /dev/pts /share/pts

> | >

> | > from the parent mntns (not that I know why you'd want to do *that* :)

> | > then the chroot'ed tasks will see the original mntns's /dev/pts under

> | > /jail1/share.

> |

> | I haven't yet tried that, but :(this function

> |

> | static inline int check_mnt(struct vfsmount *mnt)

> | {

> | return mnt->mnt_ns == current->nsproxy->mnt_ns;

> | }

> |

```

> | and this code in do_loopback
> |
> |     if (!check_mnt(nd->mnt) || !check_mnt(old_nd.mnt))
> |         goto out;
> |
> | makes me think that trying to bind a mount from another mntns
> | ot _to_ another is prohibited... Do I miss something?
> |
> | >>> Though really, I think you're right - we shouldn't break the kernel
> | >>> doing CLONE_NEWMQ or CLONE_NEWPTS without CLONE_NEWNS, so we shouldn't
> | >>> force the combination.
> | >>>
> | >>>> exploited in OpenVZ, so if we can somehow avoid forcing the NEWNS flag
> | >>>> that would be very very good :) See my next comment about this issue.
> | >>>>
> | >>>>> Pavel, not long ago you said you were starting to look at tty and pty
> | >>>>> stuff - did you have any different ideas on devpts virtualization, or
> | >>>>> are you ok with this minus your comments thus far?
> | >>>> I have a similar idea of how to implement this, but I didn't thought
> | >>>> about the details. As far as this issue is concerned, I see no reasons
> | >>>> why we need a kern_mount-ed devptsfs instance. If we don't make such,
> | >>>> we may safely hold the ptsns from the superblock and be happy. The
> | >>>> same seems applicable to the mqns, no?
> | >>> But the current->nsproxy->devpts->mnt is used in several functions in
> | >>> patch 3.
> | >> Indeed. I overlooked this. Then we're in a deep ... problem here.
> | >>
> | >> Breaking this circle was not that easy with pid namespaces, so
> | >> I put the strut in proc_flush_task - when the last task from the
> | >> namespace exits the kern-mount-ed vfsmnt is dropped, but we can't
> | >> do the same stuff with devpts.
> | >
> | > But I still don't see what the problem is with my proposal? So long as
> | > you agree that if there are no tasks remaining in the devptsns,
> | > then any task which has its devpts mounted should see an empty directory
> | > (due to sb->s_info being NULL), I think it works.
> |
> | Well, if we _do_ can handle the races with ns->devpts_mnt switch
> | from not NULL to NULL, then I'm fine with this approach.
> |
> | I just remember, that with pid namespaces this caused a complicated
> | locking and performance degradation. This is the problem I couldn't
> | remember yesterday.
>
> Well, iirc, one problem with pid namespaces was that we need to keep
> the task and pid_namespace association until the task was waited on
> (for instance the wait() call from parent needs the pid_t of the
> child which is tied to the pid ns in struct upid).

```

>
> For this reason, we don't drop the mnt reference in free_pid_ns() but
> hold the reference till proc_flush_task().
>
> With devpts, can't we simply drop the reference in free_pts_ns() so
> that when the last task using the pts_ns exits, we can unmount and
> release the mnt ?

I hope we can. The thing I'm worried about is whether we can correctly handle race with this pointer switch from NULL to not-NULL.

> IOW, do you suspect that the circular reference leads to leaking vfsmnts ?
>

Of course! If the namespace holds the vfsmnt, vfsmnt holds the superblock and the superblock holds the namespace we won't drop this chain ever, unless some other object breaks this chain.

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
