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

Posted by [Cedric Le Goater](#) on Wed, 06 Feb 2008 18:05:42 GMT

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

Serge E. Hallyn wrote:

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

>> sukadev@us.ibm.com wrote:

>>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

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

>>>

>>> To support multiple PTY namespaces, we should be allow multiple mounts of

>>> /dev/pts, once within each PTY namespace.

>>>

>>> This patch removes the get_sb_single() in devpts_get_sb() and uses test and

>>> set sb interfaces to allow remounting /dev/pts. The patch also removes the

>>> globals, 'devpts_root' and uses current_pts_mnt() to access 'devpts_mnt'

>>>

>>> Changelog:

>>> - Version 0: Based on earlier versions from Serge Hallyn and

>>> Matt Helsley.

>>>

>>> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

>>> ---

>>> fs/devpts/inode.c | 120 ++++++-----

>>> 1 file changed, 101 insertions(+), 19 deletions(-)

>>>

>>> Index: linux-2.6.24/fs/devpts/inode.c

>>> =====

>>> --- linux-2.6.24.orig/fs/devpts/inode.c 2008-02-05 17:30:52.000000000 -0800

>>> +++ linux-2.6.24/fs/devpts/inode.c 2008-02-05 19:16:39.000000000 -0800

>>> @@ -34,7 +34,10 @@ static inline struct idr *current_pts_ns

>>> }

>>>

>>> static struct vfsmount *devpts_mnt;

>>> -static struct dentry *devpts_root;

>>> +static inline struct vfsmount *current_pts_ns_mnt(void)

>>> +{

>>> + return devpts_mnt;

>>> +}

>>>

>>> static struct {

>>> int setuid;

>>> @@ -130,7 +133,7 @@ devpts_fill_super(struct super_block *s,

>>> inode->i_fop = &simple_dir_operations;

>>> inode->i_nlink = 2;

>>>

>>> - devpts_root = s->s_root = d_alloc_root(inode);

>>> + s->s_root = d_alloc_root(inode);

```

>>> if (s->s_root)
>>>     return 0;
>>>
>>> @@ -140,10 +143,53 @@ fail:
>>>     return -ENOMEM;
>>> }
>>>
>>> +/*
>>> + * We use test and set super-block operations to help determine whether we
>>> + * need a new super-block for this namespace. get_sb() walks the list of
>>> + * existing devpts supers, comparing them with the @data ptr. Since we
>>> + * passed 'current's namespace as the @data pointer we can compare the
>>> + * namespace pointer in the super-block's 's_fs_info'. If the test is
>>> + * TRUE then get_sb() returns a new active reference to the super block.
>>> + * Otherwise, it helps us build an active reference to a new one.
>>> + */
>>> +
>>> +static int devpts_test_sb(struct super_block *sb, void *data)
>>> +{
>>> + return sb->s_fs_info == data;
>>> +}
>>> +
>>> +static int devpts_set_sb(struct super_block *sb, void *data)
>>> +{
>>> + sb->s_fs_info = data;
>>> + return set_anon_super(sb, NULL);
>>> +}
>>> +
>>> static int devpts_get_sb(struct file_system_type *fs_type,
>>>     int flags, const char *dev_name, void *data, struct vfsmount *mnt)
>>> {
>>> - return get_sb_single(fs_type, flags, data, devpts_fill_super, mnt);
>>> + struct super_block *sb;
>>> + int err;
>>> +
>>> + /* hereafter we're very similar to get_sb_nodev */
>>> + sb = sget(fs_type, devpts_test_sb, devpts_set_sb, data);
>>> + if (IS_ERR(sb))
>>> +     return PTR_ERR(sb);
>>> +
>>> + if (sb->s_root)
>>> +     return simple_set_mnt(mnt, sb);
>>> +
>>> + sb->s_flags = flags;
>>> + err = devpts_fill_super(sb, data, flags & MS_SILENT ? 1 : 0);
>>> + if (err) {
>>> +     up_write(&sb->s_umount);
>>> +     deactivate_super(sb);

```

```
>>> + return err;
>>> + }
>>> +
>> That stuff becomes very very similar to that in proc :)
>> Makes sense to consolidate. Maybe...
>
> Yeah, and the mqns that Cedric sent too. I think Cedric said he'd
> started an a patch implementing a helper. Cedric?
```

yes.

it's basically a `get_sb_single_per_ns()` routine using `->s_fs_info` to distinguish the ns but there seems to be more to do to support correctly namespaces using internal filesystems (circular ref)

C.

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