Subject: Re: [RFC][PATCH 3/4]: Enable multiple mounts of /dev/pts
Posted by Sukadev Bhattiprolu on Thu, 14 Feb 2008 18:16:58 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn [serue@us.ibm.com] wrote:
|
| > 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 devtpsfs 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.

Hmm, current_pts_ns_mnt() is used in:

devpts_pty_new()
devpts_get_tty()
devpts_pty_kill()

All of these return error if current_pts_ns_mnt() returns NULL.
So, can we require user-space mount and unmount /dev/pts and return
error if any operation is attempted before the mount ?

|
| > The reason I have the kern_mount-ed instance of proc for pid namespaces
| > is that I need a vfsmount to flush task entries from, but allowing
| > it to be NULL (i.e. no kern_mount, but optional user mounts) means
| > handing all the possible races, which is too heavy. But do we actually
| > need the vfsmount for devpts and mqns if no user-space mounts exist?
| >
| > Besides, I planned to include legacy ptys virtualization and console
| > virtualizatin in this namespace, but it seems, that it is not present
| > in this particular one.
|
| I had been thinking the consoles would have their own ns, since there's
| really nothing linking them, but there really is no good reason why
| userspace should ever want them separate. So I'm fine with combining
| them.
|

```
| > >>> + sb->s_flags |= MS_ACTIVE;
| > >>> + devpts_mnt = mnt;
| > >>> +
| > >>> + return simple_set_mnt(mnt, sb);
| > >>>  }
| > >>>
| > >>>  static struct file_system_type devpts_fs_type = {
| > >>> @@ -158,10 +204,9 @@ static struct file_system_type devpts_fs
| > >>>   * to the System V naming convention
| > >>>   */
| > >>>
| > >>> -static struct dentry *get_node(int num)
| > >>> +static struct dentry *get_node(struct dentry *root, int num)
| > >>>  {
| > >>>   char s[12];
| > >>> - struct dentry *root = devpts_root;
| > >>>   mutex_lock(&root->d_inode->i_mutex);
| > >>>   return lookup_one_len(s, root, sprintf(s, "%d", num));
| > >>>  }
| > >>> @@ -207,12 +252,28 @@ int devpts_pty_new(struct tty_struct *tt
| > >>>   struct tty_driver *driver = tty->driver;
| > >>>   dev_t device = MKDEV(driver->major, driver->minor_start+number);
| > >>>   struct dentry *dentry;
| > >>> - struct inode *inode = new_inode(devpts_mnt->mnt_sb);
| > >>> + struct dentry *root;
| > >>> + struct vfsmount *mnt;
| > >>> + struct inode *inode;
| > >>> +
| > >>>
| > >>>   /* We're supposed to be given the slave end of a pty */
| > >>>   BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
| > >>>   BUG_ON(driver->subtype != PTY_TYPE_SLAVE);
| > >>>
| > >>> + mnt = current_pts_ns_mnt();
| > >>> + if (!mnt)
| > >>> +  return -ENOSYS;
| > >>> + root = mnt->mnt_root;
| > >>> +
| > >>> + mutex_lock(&root->d_inode->i_mutex);
| > >>> + inode = idr_find(current_pts_ns_allocated_ptys(), number);
| > >>> + mutex_unlock(&root->d_inode->i_mutex);
| > >>> +
| > >>> + if (inode && !IS_ERR(inode))
| > >>> +  return -EEXIST;
| > >>> +
| > >>> + inode = new_inode(mnt->mnt_sb);
| > >>>   if (!inode)
| > >>>    return -ENOMEM;
```

```
| > >>>
| > >>> @@ -222,23 +283,31 @@ int devpts_pty_new(struct tty_struct *tt
| > >>>   inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
| > >>>   init_special_inode(inode, S_IFCHR|config.mode, device);
| > >>>   inode->i_private = tty;
| > >>> + idr_replace(current_pts_ns_allocated_ptys(), inode, number);
| > >>>
| > >>> - dentry = get_node(number);
| > >>> + dentry = get_node(root, number);
| > >>>   if (!IS_ERR(dentry) && !dentry->d_inode) {
| > >>>    d_instantiate(dentry, inode);
| > >>> -  fsnotify_create(devpts_root->d_inode, dentry);
| > >>> +  fsnotify_create(root->d_inode, dentry);
| > >>>   }
| > >>>
| > >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
| > >>> + mutex_unlock(&root->d_inode->i_mutex);
| > >>>
| > >>>   return 0;
| > >>>  }
| > >>>
| > >>>  struct tty_struct *devpts_get_tty(int number)
| > >>>  {
| > >>> - struct dentry *dentry = get_node(number);
| > >>> + struct vfsmount *mnt;
| > >>> + struct dentry *dentry;
| > >>>   struct tty_struct *tty;
| > >>>
| > >>> + mnt = current_pts_ns_mnt();
| > >>> + if (!mnt)
| > >>> +  return NULL;
| > >>> +
| > >>> + dentry = get_node(mnt->mnt_root, number);
| > >>> +
| > >>>   tty = NULL;
| > >>>   if (!IS_ERR(dentry)) {
| > >>>    if (dentry->d_inode)
| > >>> @@ -246,14 +315,21 @@ struct tty_struct *devpts_get_tty(int nu
| > >>>    dput(dentry);
| > >>>   }
| > >>>
| > >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
| > >>> + mutex_unlock(&mnt->mnt_root->d_inode->i_mutex);
| > >>>
| > >>>   return tty;
| > >>>  }
| > >>>
| > >>>  void devpts_pty_kill(int number)
```

```
| > >>>  {
| > >>> - struct dentry *dentry = get_node(number);
| > >>> + struct dentry *dentry;
| > >>> + struct dentry *root;
| > >>> + struct vfsmount *mnt;
| > >>> +
| > >>> + mnt = current_pts_ns_mnt();
| > >>> + root = mnt->mnt_root;
| > >>> +
| > >>> + dentry = get_node(root, number);
| > >>>
| > >>>   if (!IS_ERR(dentry)) {
| > >>>    struct inode *inode = dentry->d_inode;
| > >>> @@ -264,17 +340,23 @@ void devpts_pty_kill(int number)
| > >>>    }
| > >>>    dput(dentry);
| > >>>   }
| > >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
| > >>> + mutex_unlock(&root->d_inode->i_mutex);
| > >>>  }
| > >>>
| > >>>  static int __init init_devpts_fs(void)
| > >>>  {
| > >>> - int err = register_filesystem(&devpts_fs_type);
| > >>> - if (!err) {
| > >>> -  devpts_mnt = kern_mount(&devpts_fs_type);
| > >>> -  if (IS_ERR(devpts_mnt))
| > >>> -   err = PTR_ERR(devpts_mnt);
| > >>> - }
| > >>> + struct vfsmount *mnt;
| > >>> + int err;
| > >>> +
| > >>> + err = register_filesystem(&devpts_fs_type);
| > >>> + if (err)
| > >>> +  return err;
| > >>> +
| > >>> + mnt = kern_mount_data(&devpts_fs_type, NULL);
| > >>> + if (IS_ERR(mnt))
| > >>> +  err = PTR_ERR(mnt);
| > >>> + else
| > >>> +  devpts_mnt = mnt;
| > >>>   return err;
| > >>>  }
| > >>>
| > >>> _____________________________________________
| > >>> Containers mailing list
| > >>> Containers@lists.linux-foundation.org
| > >>> https://lists.linux-foundation.org/mailman/listinfo/containers
```

| > >>>
| > >>> _____
| > >>> Devel mailing list
| > >>> Devel@openvz.org
| > >>> https://openvz.org/mailman/listinfo/devel
| > >>>
| > >> _____
| > >> Containers mailing list
| > >> Containers@lists.linux-foundation.org
| > >> https://lists.linux-foundation.org/mailman/listinfo/containers
| > >

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