

---

Subject: Re: How to query mount propagation state?  
Posted by [Ram Pai](#) on Tue, 17 Apr 2007 06:55:31 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Mon, 2007-04-16 at 14:16 -0500, Serge E. Hallyn wrote:

> > This patch introduces a new proc interface that exposes all the  
> propagation  
> > trees within the namespace.  
> >  
> > It walks through each off the mounts in the namespace, and prints  
> the following information.  
> >  
> > mount-id: a unique mount identifier  
> > dev-id : the unique device used to identify the device containing  
> the filesystem  
> > path-from-root: mount point of the mount from /  
> > path-from-root-of-its-sb: path from its own root dentry.  
> > propagation-flag: SHARED, SLAVE, UNBINDABLE, PRIVATE  
> > peer-mount-id: the mount-id of its peer mount (if this mount is  
> shared)  
> > master-mount-id: the mount-id of its master mount (if this mount is  
> slave)  
> >  
> > Using the above information one could easily write a script that can  
> > draw all the propagation trees in the namespace.  
> >  
> >  
> > Example:  
> > Here is a sample output of cat /proc/\$\$/mounts\_propagation  
> >  
> > 0xa917800 0x1 // PRIVATE  
> > 0xa917200 0x6200 // PRIVATE  
> > 0xa917180 0x3 /proc / PRIVATE  
> > 0xa917f80 0xa /dev/pts / PRIVATE  
> > 0xa917100 0x6210 /mnt / SHARED peer:0xa917100  
> > 0xa917f00 0x6210 /tmp /1 SLAVE master:0xa917100  
> > 0xa917900 0x6220 /mnt/2 / SHARED peer:0xa917900  
> >  
> > line 5 indicates that the mount with id 0xa917100 is mounted at /mnt  
> is shared  
> > and it is the only mount in its peer group.  
> >  
> > line 6 indicates that the mount with id 0xa917f00 is mounted  
> at /tmp, its  
> > root is the dentry 1 present under its root directory. This mount is  
> a  
> > slave mount and its master is the mount with id 0xa917100.  
> >

```

> > line 7 indicates that the mount with id 0xa917900 is mounted
> > at /mnt/2, its
> > root is the dentry / of its filesystem. This mount is a
> > shared and it is the only mount in its peer group.
> >
> > one could write a script which runs through these lines and draws 4
> > individual satellite mounts and two propagation trees, the first
> > propagation
> > tree has a shared mount and a slave mount. and the second
> > propagation tree has
> > just one shared mount.
> >
> >
> > Signed-off-by: Ram Pai <linuxram@us.ibm.com>
> > ---
> > fs/namespace.c | 42 +++++
> > fs/pnode.c      | 6 -----
> > fs/pnode.h      | 6 +++++
> > fs/proc/base.c | 22 +++++
> > 4 files changed, 69 insertions(+), 7 deletions(-)
> >
> > Index: linux-2.6.17.10/fs/namespace.c
> > =====
> > --- linux-2.6.17.10.orig/fs/namespace.c
> > +++ linux-2.6.17.10/fs/namespace.c
> > @@ -410,6 +410,41 @@ static int show_vfsmnt_new(struct seq_file
> >     return show_options(m, v);
> > }
> >
> > +static int show_vfsmnt_propagation(struct seq_file *m, void *v)
> > +{
> > +    struct vfsmount *mnt = v;
> > +    seq_printf(m, "0x%x", (int)mnt);
> > +    seq_putc(m, ' ');
> > +    seq_printf(m, "0x%x", new_encode_dev(mnt->mnt_sb->s_dev));
> > +    seq_putc(m, ' ');
> > +    seq_path(m, mnt, mnt->mnt_root, " \t\n\\");
> > +    seq_putc(m, ' ');
> > +    seq_dentry(m, mnt->mnt_root, " \t\n\\");
> > +    seq_putc(m, ' ');
> > +
> > +    if (IS_MNT_SHARED(mnt)) {
> > +        seq_printf(m, "%s ", "SHARED");
> > +        if (IS_MNT_SLAVE(mnt)) {
> > +            seq_printf(m, "%s ", "SLAVE");
> > +        }
> > +    } else if (IS_MNT_SLAVE(mnt)) {
> > +        seq_printf(m, "%s ", "SLAVE");

```

```

>>+ } else if (IS_MNT_UNBINDABLE(mnt)) {
>>+     seq_printf(m, "%s ", "UNBINDABLE");
>>+ } else {
>>+     seq_printf(m, "%s ", "PRIVATE");
>>+ }
>>+
>>+ if (IS_MNT_SHARED(mnt)) {
>>+     seq_printf(m, "peer:0x%x ", (int)next_peer(mnt));
>
> Ok, so if the sequence of events was
>
>     mount --make-shared /mnt
>     (some user logs in and gets a cloned namespace, so his /mnt
>     becomes the next peer of /mnt)
>     mount --bind /mnt /tmp
>     (some other user logs in and gets cloned namespace...)
>
> or some such sequence of events, we could lose all information
> about /mnt and /tmp being peers, right? Should a new
> next_peer_in_same_namespace(mnt) be used rather than next_peer()?

```

you are right. it should print next\_peer(mnt) only if CAP\_SYS\_ADMIN, else print next\_peer\_in\_same\_namespace(mnt).

```

>
> Somewhat similarly,
>
>>+ }
>>+ if (IS_MNT_SLAVE(mnt)) {
>>+     seq_printf(m, "master:0x%x ", (int)mnt->mnt_master);
>
> Should we for privacy reasons not print out the address
> mnt->mnt_master
> is in a different namespace (perhaps if !CAP_SYS_ADMIN)?

```

right. it should print mnt->mnt\_master if (CAP\_SYS\_ADMIN), otherwise print master\_in\_same\_namespace(mnt).

RP

```

>
> Otherwise I like this.
>
> thanks,
> -serge

```

---

Containers mailing list

