
Subject: [PATCH 00/25] Mount writer count and read-only bind mounts (v7)

Posted by [Dave Hansen](#) on Mon, 11 Dec 2006 22:30:00 GMT

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

I figured I'd run this past the containers list one more time.

These are against 2.6.19-mm1. They compile, boot and pass my little attached test script.

Changes from previous version

- rework helper names for doing custom 'struct file' creation

The following series implements read-only bind mounts. This feature allows a read-only view into a read-write filesystem. In the process of doing that, it also provides infrastructure for keeping track of the number of writers to any given mount. In this version, if there are writers on a superblock, the filesystem may not be remounted r/o. The same goes for MS_BIND mounts, and writers on a vfsmount.

This has a number of uses. It allows chroots to have parts of filesystems writable. It will be useful for containers in the future and is intended to replace patches that vserver has had out of the tree for several years. It allows security enhancement by making sure that parts of your filesystem read-only, when you don't want to have entire new filesystems mounted, or when you want atime selectively updated.

This set makes no attempt to keep the return codes for these r/o bind mounts the same as for a real r/o filesystem or device. It would require significantly more code and be quite a bit more invasive.

Using this feature requires two steps:

```
mount --bind /source /dest
mount -o remount,ro /dest
```

I've been using the following script to test that the feature is working as desired. It takes a directory and makes a regular bind and a r/o bind mount of it. It then performs some normal filesystem operations on the three directories, including ones that are expected to fail, like creating a file on the r/o mount.

```
#!/bin/sh
DIRS="foo foo-bound foo-bound-ro"
```

```

set -e

function do_umount
{
    (
        umount foo-bound || true;
        umount foo-bound-ro || true;
    ) 2> /dev/null
}

trap do_umount ERR

# just in case the last invocation left them
do_umount

function should_fail
{
    "$@" > /dev/null 2>&1 \
    && echo unexpected success: "$@" \
    || echo GOOD: expected failure: "$@";
}
function should_succeed
{
    "$@" > /dev/null 2>&1 \
    && echo GOOD: expected success: "$@" \
    || echo unexpected failure: "$@"
}
function should_fail_ro
{
    RO=$1
    shift
    if $RO; then
        should_fail "$@"
    else
        should_succeed "$@"
    fi
}
function testdir
{
    RO=$1
    shift;
    i=$1
    should_fail_ro $RO touch $i/$i-file
    should_fail_ro $RO mkdir $i/$i-dir
    should_fail_ro $RO mknod $i/$i-null-chardev c 1 3
    should_fail_ro $RO chmod 777 $i/$i-null-chardev
}
for i in $DIRS; do

```

```
rm -r ./${i} > /dev/null 2>&1 || true
mkdir -p ${i};
done;
```

```
mount --bind foo foo-bound/ || exit
mount --bind foo foo-bound-ro || exit
mount -o remount,ro foo-bound-ro || exit
```

```
testdir false foo
testdir false foo-bound
testdir true foo-bound-ro
```

```
should_succeed echo foo > foo-bound/foo-null-chardev
should_succeed echo foo > foo-bound-ro/foo-null-chardev
```

```
should_fail chmod 777 foo-bound-ro/foo-dir
should_fail chmod 777 foo-bound-ro/foo-file
should_fail chown nobody foo-bound-ro/foo-dir
should_fail chown nobody foo-bound-ro/foo-file
should_fail rmdir foo-bound-ro/foo-dir
```

```
do_umount
```

Signed-off-by: Dave Hansen <haveblue@us.ibm.com>

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
