Subject: [PATCH 4/4]: Enable multiple mounts of /dev/pts
Posted by Sukadev Bhattiprolu on Sat, 12 Apr 2008 17:34:28 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject:[PATCH 4/4]: Enable multiple mounts of /dev/pts

To support multiple PTY namespaces, 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.

Changelog [v4]:
 - Split-off the simpler changes of moving global=variables into
   'pts_namespace' to previous patch.

Changelog [v3]:
 - Removed some unnecessary comments from devpts_set_sb()

Changelog [v2]:

 - (Pavel Emelianov/Serge Hallyn) Remove reference to pts_ns from
   sb->s_fs_info to fix the circular reference (/dev/pts is not
   unmounted unless the pts_ns is destroyed, so we don't need a
   reference to the pts_ns).

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Serge Hallyn <serue@us.ibm.com>
Signed-off-by: Matt Helsley <matthltc@us.ibm.com>
---
 fs/devpts/inode.c | 62 ++---
 1 file changed, 59 insertions(+), 3 deletions(-)

Index: 2.6.25-rc8-mm1/fs/devpts/inode.c
===
--- 2.6.25-rc8-mm1.orig/fs/devpts/inode.c 2008-04-12 10:10:33.000000000 -0700
+++ 2.6.25-rc8-mm1/fs/devpts/inode.c 2008-04-12 10:10:38.000000000 -0700
@@ -154,17 +154,73 @@ 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;
+ struct pts_namespace *ns;
+ int err;
+
+ /* hereafter we're very similar to proc_get_sb */
+ if (flags & MS_KERNMOUNT)
+  ns = data;
+ else
+  ns = &init_pts_ns;
+
+ /* hereafter we're very simlar to get_sb_nodev */
+ sb = sget(fs_type, devpts_test_sb, devpts_set_sb, ns);
+ 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;
+ }
+
+ sb->s_flags |= MS_ACTIVE;
+ ns->mnt = mnt;
+
```

```
+ return simple_set_mnt(mnt, sb);
+}
+
+static void devpts_kill_sb(struct super_block *sb)
+{
+ sb->s_fs_info = NULL;
+ kill_anon_super(sb);
 }

 static struct file_system_type devpts_fs_type = {
  .owner  = THIS_MODULE,
  .name  = "devpts",
  .get_sb  = devpts_get_sb,
- .kill_sb = kill_anon_super,
+ .kill_sb = devpts_kill_sb,
 };

 /*
@@ -315,7 +371,7 @@ static int __init init_devpts_fs(void)

  err = register_filesystem(&devpts_fs_type);
  if (!err) {
-  mnt = kern_mount(&devpts_fs_type);
+  mnt = kern_mount_data(&devpts_fs_type, &init_pts_ns);
   if (IS_ERR(mnt))
    err = PTR_ERR(mnt);
   else
```