Subject: [PATCH 0/7][v2] Clone PTY namespaces
Posted by Sukadev Bhattiprolu on Thu, 03 Apr 2008 07:02:33 GMT
View Forum Message <> Reply to Message

Devpts namespace patchset

In continuation of the implementation of containers in mainline, we need to support multiple PTY namespaces so that the PTY index (ie the tty names) in one container is independent of the PTY indices of other containers. For instance this would allow each container to have a '/dev/pts/0' PTY and refer to different terminals.

[PATCH 1/7]: Propagate error code from devpts_pty_new
[PATCH 2/7]: Factor out PTY index allocation
[PATCH 3/7]: Enable multiple mounts of /dev/pts
[PATCH 4/7]: Allow mknod of ptmx and tty in devpts
[PATCH 5/7]: Implement get_pts_ns() and put_pts_ns()
[PATCH 6/7]: Determine pts_ns from a pty's inode
[PATCH 7/7]: Enable cloning PTY namespaces

Todo:
- This patchset depends on availability of additional clone flags. and relies on on Cedric's clone64 patchset.

- Needs some cleanup and more testing.

- Ensure patchset is bisect-safe

Changelog[v2]:

(Patches 4 and 6 differ significantly from [v1]. Others are mostly the same)

- [Alexey Dobriyan, Pavel Emelyanov] Removed the hack to check for user-space mount.

- [Serge Hallyn] Added rcu locking around access to sb->s_fs_info.

- [Serge Hallyn] Allow creation of /dev/pts/ptmx and /dev/pts/tty devices to simplify the process of finding the 'owning' pts-ns of the device (specially when accessed from parent-pts-ns) See patches 4 and 6 for details.

Changelog[v1]:
- Fixed circular reference by not caching the pts_ns in sb->s_fs_info (without incrementing reference count) and clearing the sb->s_fs_info when destroying the pts_ns

- To allow access to a child container's ptys from parent container,
  determine the 'pts_ns' of a 'pty' from its inode.

- Added a check (hack) to ensure user-space mount of /dev/pts is
  done before creating PTYs in a new pts-ns.

- Reorganized the patchset and removed redundant changes.

- Ported to work wih Cedric Le Goater's clone64() system call now
  that we are out of clone_flags.

Changelog[v0]:

This patchset is based on earlier versions developed by Serge Hallyn
and Matt Helsley.

_____

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

---

Subject: [PATCH 3/7][v2]: Enable multiple mounts of /dev/pts
Posted by Sukadev Bhattiprolu on Thu, 03 Apr 2008 07:09:04 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH 3/7][v2]: 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_mnt', 'devpts_root'
and uses a skeletal 'init_pts_ns' to store the vfsmount.

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         |  151 ++++++++++++++++++++++++++++++++---------
 include/linux/devpts_fs.h |   11 +++
 2 files changed, 134 insertions(+), 28 deletions(-)

Index: 2.6.25-rc5-mm1/include/linux/devpts_fs.h
===================================================================
--- 2.6.25-rc5-mm1.orig/include/linux/devpts_fs.h 2008-03-24 20:04:26.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-04-01 18:08:42.000000000 -0700
@@ -14,6 +14,17 @@
 #define _LINUX_DEVPTS_FS_H

 #include <linux/errno.h>
+#include <linux/nsproxy.h>
+#include <linux/kref.h>
+#include <linux/idr.h>
+
+struct pts_namespace {
+ struct kref kref;
+ struct idr allocated_ptys;
+ struct vfsmount *mnt;
+};
+
+extern struct pts_namespace init_pts_ns;

 #ifdef CONFIG_UNIX98_PTYS

Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-03-24 20:04:26.000000000 -0700
+++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-04-01 18:08:41.000000000 -0700
@@ -28,12 +28,8 @@
 #define DEVPTS_DEFAULT_MODE 0600

 extern int pty_limit;   /* Config limit on Unix98 ptys */
-static DEFINE_IDR(allocated_ptys);
 static DECLARE_MUTEX(allocated_ptys_lock);

-static struct vfsmount *devpts_mnt;
-static struct dentry *devpts_root;
-
 static struct {
  int setuid;
  int setgid;
@@ -54,6 +50,15 @@ static match_table_t tokens = {
  {Opt_err, NULL}
 };
```

```
+struct pts_namespace init_pts_ns = {
+ .kref = {
+  .refcount = ATOMIC_INIT(2),
+ },
+ .allocated_ptys = IDR_INIT(init_pts_ns.allocated_ptys),
+ .mnt = NULL,
+};
+
+
 static int devpts_remount(struct super_block *sb, int *flags, char *data)
 {
  char *p;
@@ -140,7 +145,7 @@ 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;

@@ -150,17 +155,73 @@ 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,
 };
```

```
 /*
@@ -168,10 +229,9 @@ 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));
 }
@@ -180,14 +240,15 @@ int devpts_new_index(void)
 {
  int index;
  int idr_ret;
+ struct pts_namespace *pts_ns = &init_pts_ns;

 retry:
- if (!idr_pre_get(&allocated_ptys, GFP_KERNEL)) {
+ if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
   return -ENOMEM;
  }

  down(&allocated_ptys_lock);
- idr_ret = idr_get_new(&allocated_ptys, NULL, &index);
+ idr_ret = idr_get_new(&pts_ns->allocated_ptys, NULL, &index);
  if (idr_ret < 0) {
   up(&allocated_ptys_lock);
   if (idr_ret == -EAGAIN)
@@ -196,7 +257,7 @@ retry:
  }

  if (index >= pty_limit) {
-  idr_remove(&allocated_ptys, index);
+  idr_remove(&pts_ns->allocated_ptys, index);
   up(&allocated_ptys_lock);
   return -EIO;
  }
@@ -206,8 +267,10 @@ retry:

 void devpts_kill_index(int idx)
 {
+ struct pts_namespace *pts_ns = &init_pts_ns;
+
  down(&allocated_ptys_lock);
```

```
- idr_remove(&allocated_ptys, idx);
+ idr_remove(&pts_ns->allocated_ptys, idx);
  up(&allocated_ptys_lock);
 }

@@ -217,12 +280,26 @@ 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;
+ struct pts_namespace *pts_ns = &init_pts_ns;

  /* 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 = pts_ns->mnt;
+ root = mnt->mnt_root;
+
+ mutex_lock(&root->d_inode->i_mutex);
+ inode = idr_find(&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;

@@ -232,23 +309,29 @@ 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(&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 = init_pts_ns.mnt;
+
+ dentry = get_node(mnt->mnt_root, number);
+
  tty = NULL;
  if (!IS_ERR(dentry)) {
   if (dentry->d_inode)
@@ -256,14 +339,19 @@ 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;
+
+ root = init_pts_ns.mnt->mnt_root;
+
+ dentry = get_node(root, number);

  if (!IS_ERR(dentry)) {
   struct inode *inode = dentry->d_inode;
@@ -274,24 +362,31 @@ 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, &init_pts_ns);
+ if (IS_ERR(mnt))
+  err = PTR_ERR(mnt);
+ else
+  init_pts_ns.mnt = mnt;
  return err;
 }

 static void __exit exit_devpts_fs(void)
 {
  unregister_filesystem(&devpts_fs_type);
- mntput(devpts_mnt);
+ mntput(init_pts_ns.mnt);
+ init_pts_ns.mnt = NULL;
 }

 module_init(init_devpts_fs)
```

Subject: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode
Posted by Sukadev Bhattiprolu on Thu, 03 Apr 2008 07:11:07 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode.

The devpts interfaces currently operate on a specific pts namespace
which they get from the 'current' task.

With implementation of containers and cloning of PTS namespaces, we want
to be able to access PTYs in a child-pts-ns from a parent-pts-ns. For
instance we could bind-mount and pivot-root the child container on
'/vserver/vserver1' and then access the "pts/0" of 'vserver1' using

 $ echo foo > /vserver/vserver1/dev/pts/0

The task doing the above 'echo' could be in parent-pts-ns. So we find
the 'pts-ns' of the above file from the inode representing the above
file rather than from the 'current' task.

Note that we need to find and hold a reference to the pts_ns to prevent
the pts_ns from being freed while it is being accessed from 'outside'.

This patch implements, 'pts_ns_from_inode()' which returns the pts_ns
using 'inode->i_sb->s_fs_info'.

Since, the 'inode' information is not visible inside devpts code itself,
this patch modifies the tty driver code to determine the pts_ns and passes
it into devpts.

Changelog [v2]:
 [Serge Hallyn] Use rcu to access sb->s_fs_info.

 [Serge Hallyn] Simplify handling of ptmx and tty devices by expecting
  user to create them in /dev/pts (see also devpts-mknod patch)

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 drivers/char/pty.c | 13 +++++-
 drivers/char/tty_io.c | 93 ++++++++++++++++++++++++++++++++++++++-------
 fs/devpts/inode.c | 19 +++++----
 include/linux/devpts_fs.h | 38 +++++++++++++++--
 4 files changed, 131 insertions(+), 32 deletions(-)

Index: 2.6.25-rc5-mm1/include/linux/devpts_fs.h
===
--- 2.6.25-rc5-mm1.orig/include/linux/devpts_fs.h 2008-04-02 22:42:08.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-04-02 22:42:14.000000000 -0700
@@ -17,6 +17,7 @@
 #include <linux/nsproxy.h>
 #include <linux/kref.h>
 #include <linux/idr.h>
+#include <linux/fs.h>

 struct pts_namespace {

```
  struct kref kref;
@@ -26,12 +27,39 @@ struct pts_namespace {

 extern struct pts_namespace init_pts_ns;

+#define DEVPTS_SUPER_MAGIC  0x1cd1
+
+static inline struct pts_namespace *current_pts_ns(void)
+{
+ return &init_pts_ns;
+}
+
+static inline struct pts_namespace *pts_ns_from_inode(struct inode *inode)
+{
+ /*
+  * If this file exists on devpts, return the pts_ns from the
+  * devpts super-block. Otherwise just use the pts-ns of the
+  * calling task.
+  */
+ if(inode->i_sb->s_magic == DEVPTS_SUPER_MAGIC)
+  return rcu_dereference(inode->i_sb->s_fs_info);
+
+ return current_pts_ns();
+}
+
+
 #ifdef CONFIG_UNIX98_PTYS
-int devpts_new_index(void);
-void devpts_kill_index(int idx);
-int devpts_pty_new(struct tty_struct *tty);      /* mknod in devpts */
-struct tty_struct *devpts_get_tty(int number);  /* get tty structure */
-void devpts_pty_kill(int number);   /* unlink */
+int devpts_new_index(struct pts_namespace *pts_ns);
+void devpts_kill_index(struct pts_namespace *pts_ns, int idx);
+
+/* mknod in devpts */
+int devpts_pty_new(struct pts_namespace *pts_ns, struct tty_struct *tty);
+
+/* get tty structure */
+struct tty_struct *devpts_get_tty(struct pts_namespace *pts_ns, int number);
+
+/* unlink */
+void devpts_pty_kill(struct pts_namespace *pts_ns, int number);

 static inline void free_pts_ns(struct kref *ns_kref) { }

Index: 2.6.25-rc5-mm1/drivers/char/tty_io.c
===================================================================
```

```
--- 2.6.25-rc5-mm1.orig/drivers/char/tty_io.c 2008-04-02 22:35:29.000000000 -0700
+++ 2.6.25-rc5-mm1/drivers/char/tty_io.c 2008-04-02 22:42:14.000000000 -0700
@@ -2064,8 +2064,8 @@ static void tty_line_name(struct tty_dri
  * relaxed for the (most common) case of reopening a tty.
  */

-static int init_dev(struct tty_driver *driver, int idx,
- struct tty_struct **ret_tty)
+static int init_dev(struct tty_driver *driver, struct pts_namespace *pts_ns,
+  int idx, struct tty_struct **ret_tty)
 {
  struct tty_struct *tty, *o_tty;
  struct ktermios *tp, **tp_loc, *o_tp, **o_tp_loc;
@@ -2074,7 +2074,11 @@ static int init_dev(struct tty_driver *d

  /* check whether we're reopening an existing tty */
  if (driver->flags & TTY_DRIVER_DEVPTS_MEM) {
-  tty = devpts_get_tty(idx);
+  tty = devpts_get_tty(pts_ns, idx);
+  if (IS_ERR(tty)) {
+   retval = PTR_ERR(tty);
+   goto end_init;
+  }
   /*
    * If we don't have a tty here on a slave open, it's because
    * the master already started the close process and there's
@@ -2361,6 +2365,31 @@ static void release_tty(struct tty_struc
 }

 /*
+ * When opening /dev/tty and /dev/ptmx, use the pts-ns of the calling
+ * process. For any other pts device, use the pts-ns, in which the
+ * device was created. This latter case is needed when the pty is
+ * being accessed from a parent container.
+ *
+ * Eg:  Suppose the user used bind-mount and pivot-root to mount a
+ *  child- container's root on /vs/vs1. Then "/vs/vs1/dev/pts/0"
+ *  in parent container and "/dev/pts/0" in child container would
+ *  refer to the same device.
+ *
+ *  When parent-container opens, "/vs/vs1/dev/pts/0" we find and
+ *  grab/drop reference to child container's pts-ns (using @filp).
+ */
+struct pts_namespace *pty_pts_ns(struct tty_driver *driver, struct inode *inode)
+{
+ struct pts_namespace *pts_ns;
+
+ pts_ns = NULL;
```

```
+ if (driver->flags & TTY_DRIVER_DEVPTS_MEM)
+  pts_ns = pts_ns_from_inode(inode);
+
+ return pts_ns;
+}
+
+/*
  * Even releasing the tty structures is a tricky business.. We have
  * to be very careful that the structures are all released at the
  * same time, as interrupts might otherwise get the wrong pointers.
@@ -2376,10 +2405,12 @@ static void release_dev(struct file *fil
  int idx;
  char buf[64];
  unsigned long flags;
+ struct pts_namespace *pts_ns;
+ struct inode *inode;

+ inode = filp->f_path.dentry->d_inode;
  tty = (struct tty_struct *)filp->private_data;
- if (tty_paranoia_check(tty, filp->f_path.dentry->d_inode,
-       "release_dev"))
+ if (tty_paranoia_check(tty, inode, "release_dev"))
   return;

  check_tty_count(tty, "release_dev");
@@ -2392,6 +2423,12 @@ static void release_dev(struct file *fil
  devpts = (tty->driver->flags & TTY_DRIVER_DEVPTS_MEM) != 0;
  o_tty = tty->link;

+ /*
+  * We already have a reference to pts_ns here, so it cannot
+  * be going away.
+  */
+ pts_ns = pty_pts_ns(tty->driver, inode);
+
 #ifdef TTY_PARANOIA_CHECK
  if (idx < 0 || idx >= tty->driver->num) {
   printk(KERN_DEBUG "release_dev: bad idx when trying to "
@@ -2569,6 +2606,10 @@ static void release_dev(struct file *fil

  mutex_unlock(&tty_mutex);

+ /* drop the reference from ptmx_open/tty_open() */
+ if (devpts)
+  put_pts_ns(pts_ns);
+
  /* check whether both sides are closing ... */
  if (!tty_closing || (o_tty && !o_tty_closing))
```

```
   return;
@@ -2634,7 +2675,7 @@ static void release_dev(struct file *fil

  /* Make this pty number available for reallocation */
  if (devpts)
-  devpts_kill_index(idx);
+  devpts_kill_index(pts_ns, idx);
 }

 /**
@@ -2666,6 +2707,7 @@ static int tty_open(struct inode *inode,
  int index;
  dev_t device = inode->i_rdev;
  unsigned short saved_flags = filp->f_flags;
+ struct pts_namespace *pts_ns;

  nonseekable_open(inode, filp);

@@ -2715,10 +2757,27 @@ retry_open:
   return -ENODEV;
  }
 got_driver:
- retval = init_dev(driver, index, &tty);
+
+ /*
+  * What pts-ns do we want to use when opening "/dev/tty" ?
+  * Sounds like current_pts_ns(), but what should happen
+  * if parent pts ns does:
+  *
+  *  echo foo > /vs/vs1/dev/tty
+  *
+  * (See Serge's setupvs1 script for the /vs/vs1...)
+  */
+ rcu_read_lock();
+ pts_ns = pty_pts_ns(driver, inode);
+ get_pts_ns(pts_ns);
+ rcu_read_unlock();
+
+ retval = init_dev(driver, pts_ns, index, &tty);
  mutex_unlock(&tty_mutex);
- if (retval)
+ if (retval) {
+  put_pts_ns(pts_ns);
   return retval;
+ }

  filp->private_data = tty;
  file_move(filp, &tty->tty_files);
```

```
@@ -2790,16 +2849,22 @@ static int ptmx_open(struct inode *inode
  struct tty_struct *tty;
  int retval;
  int index;
+ struct pts_namespace *pts_ns;

  nonseekable_open(inode, filp);

+ rcu_read_lock();
+ pts_ns = pts_ns_from_inode(inode);
+ get_pts_ns(pts_ns);
+ rcu_read_unlock();
+
  /* find a device that is not in use. */
- index = devpts_new_index();
+ retval = index = devpts_new_index(pts_ns);
  if (index < 0)
-  return index;
+  goto drop_ns;

  mutex_lock(&tty_mutex);
- retval = init_dev(ptm_driver, index, &tty);
+ retval = init_dev(ptm_driver, pts_ns, index, &tty);
  mutex_unlock(&tty_mutex);

  if (retval)
@@ -2809,7 +2874,7 @@ static int ptmx_open(struct inode *inode
  filp->private_data = tty;
  file_move(filp, &tty->tty_files);

- retval = devpts_pty_new(tty->link);
+ retval = devpts_pty_new(pts_ns, tty->link);
  if (retval)
   goto out1;

@@ -2821,7 +2886,9 @@ out1:
  release_dev(filp);
  return retval;
 out:
- devpts_kill_index(index);
+ devpts_kill_index(pts_ns, index);
+drop_ns:
+ put_pts_ns(pts_ns);
  return retval;
 }
 #endif
Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
===================================================================
```

```
--- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-04-02 22:42:01.000000000 -0700
+++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-04-02 22:42:14.000000000 -0700
@@ -23,8 +23,6 @@
 #include <linux/fsnotify.h>
 #include <linux/seq_file.h>

-#define DEVPTS_SUPER_MAGIC 0x1cd1
-
 #define DEVPTS_DEFAULT_MODE 0600

 extern int pty_limit;   /* Config limit on Unix98 ptys */
@@ -283,11 +281,10 @@ static struct dentry *get_node(struct de
  return lookup_one_len(s, root, sprintf(s, "%d", num));
 }

-int devpts_new_index(void)
+int devpts_new_index(struct pts_namespace *pts_ns)
 {
  int index;
  int idr_ret;
- struct pts_namespace *pts_ns = &init_pts_ns;

 retry:
  if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
@@ -312,16 +309,15 @@ retry:
  return index;
 }

-void devpts_kill_index(int idx)
+void devpts_kill_index(struct pts_namespace *pts_ns, int idx)
 {
- struct pts_namespace *pts_ns = &init_pts_ns;

  down(&allocated_ptys_lock);
  idr_remove(&pts_ns->allocated_ptys, idx);
  up(&allocated_ptys_lock);
 }

-int devpts_pty_new(struct tty_struct *tty)
+int devpts_pty_new( struct pts_namespace *pts_ns, struct tty_struct *tty)
 {
  int number = tty->index; /* tty layer puts index from devpts_new_index() in here */
  struct tty_driver *driver = tty->driver;
@@ -330,7 +326,6 @@ int devpts_pty_new(struct tty_struct *tt
  struct dentry *root;
  struct vfsmount *mnt;
  struct inode *inode;
- struct pts_namespace *pts_ns = &init_pts_ns;
```

```
  /* We're supposed to be given the slave end of a pty */
  BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
@@ -369,13 +364,13 @@ int devpts_pty_new(struct tty_struct *tt
  return 0;
 }

-struct tty_struct *devpts_get_tty(int number)
+struct tty_struct *devpts_get_tty(struct pts_namespace *pts_ns, int number)
 {
  struct vfsmount *mnt;
  struct dentry *dentry;
  struct tty_struct *tty;

- mnt = init_pts_ns.mnt;
+ mnt = pts_ns->mnt;

  dentry = get_node(mnt->mnt_root, number);

@@ -391,12 +386,12 @@ struct tty_struct *devpts_get_tty(int nu
  return tty;
 }

-void devpts_pty_kill(int number)
+void devpts_pty_kill(struct pts_namespace *pts_ns, int number)
 {
  struct dentry *dentry;
  struct dentry *root;

- root = init_pts_ns.mnt->mnt_root;
+ root = pts_ns->mnt->mnt_root;

  dentry = get_node(root, number);

Index: 2.6.25-rc5-mm1/drivers/char/pty.c
===================================================================
--- 2.6.25-rc5-mm1.orig/drivers/char/pty.c 2008-04-02 22:35:29.000000000 -0700
+++ 2.6.25-rc5-mm1/drivers/char/pty.c 2008-04-02 22:42:14.000000000 -0700
@@ -37,6 +37,9 @@ static struct tty_driver *pts_driver;

 static void pty_close(struct tty_struct * tty, struct file * filp)
 {
+ struct inode *inode;
+ struct pts_namespace *pts_ns;
+
  if (!tty)
   return;
  if (tty->driver->subtype == PTY_TYPE_MASTER) {
```

```
@@ -58,8 +61,14 @@ static void pty_close(struct tty_struct
  if (tty->driver->subtype == PTY_TYPE_MASTER) {
   set_bit(TTY_OTHER_CLOSED, &tty->flags);
 #ifdef CONFIG_UNIX98_PTYS
-  if (tty->driver == ptm_driver)
-   devpts_pty_kill(tty->index);
+  if (tty->driver == ptm_driver) {
+   inode = filp->f_path.dentry->d_inode;
+   rcu_read_lock();
+   pts_ns = pts_ns_from_inode(inode);
+   rcu_read_unlock();
+
+   devpts_pty_kill(pts_ns, tty->index);
+  }
 #endif
   tty_vhangup(tty->link);
  }
```

_____

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

---

Subject: [PATCH 7/7][v2]: Enable cloning PTY namespaces
Posted by Sukadev Bhattiprolu on Thu, 03 Apr 2008 07:11:44 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH 7/7][v2]: Enable cloning PTY namespaces

Enable cloning PTY namespaces.

Changelog[v2]:
 [Serge Hallyn]: Use rcu to access sb->s_fs_info.

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        |   84 ++++++++++++++++++++++++++++++++++++++++++++++--
 include/linux/devpts_fs.h |   22 ++++++++++--
 include/linux/init_task.h |    1
 include/linux/nsproxy.h   |    2 +
 include/linux/sched.h     |    1
 kernel/fork.c             |    2 -
 kernel/nsproxy.c          |   17 ++++++++-
 7 files changed, 122 insertions(+), 7 deletions(-)
```

```
Index: 2.6.25-rc5-mm1/include/linux/sched.h
===================================================================
--- 2.6.25-rc5-mm1.orig/include/linux/sched.h 2008-04-02 22:50:22.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/sched.h 2008-04-02 22:51:59.000000000 -0700
@@ -28,6 +28,7 @@
 #define CLONE_NEWPID  0x20000000 /* New pid namespace */
 #define CLONE_NEWNET  0x40000000 /* New network namespace */
 #define CLONE_IO  0x80000000 /* Clone io context */
+#define CLONE_NEWPTS  0x0000000200000000ULL /* Clone pts ns */

 /*
  * Scheduling policies
Index: 2.6.25-rc5-mm1/include/linux/nsproxy.h
===================================================================
--- 2.6.25-rc5-mm1.orig/include/linux/nsproxy.h 2008-04-02 22:50:22.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/nsproxy.h 2008-04-02 22:51:59.000000000 -0700
@@ -8,6 +8,7 @@ struct mnt_namespace;
 struct uts_namespace;
 struct ipc_namespace;
 struct pid_namespace;
+struct pts_namespace;

 /*
  * A structure to contain pointers to all per-process
@@ -29,6 +30,7 @@ struct nsproxy {
  struct pid_namespace *pid_ns;
  struct user_namespace *user_ns;
  struct net       *net_ns;
+ struct pts_namespace *pts_ns;
 };
 extern struct nsproxy init_nsproxy;

Index: 2.6.25-rc5-mm1/include/linux/init_task.h
===================================================================
--- 2.6.25-rc5-mm1.orig/include/linux/init_task.h 2008-04-02 22:50:22.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/init_task.h 2008-04-02 22:51:59.000000000 -0700
@@ -78,6 +78,7 @@ extern struct nsproxy init_nsproxy;
  .mnt_ns  = NULL,      \
  INIT_NET_NS(net_ns)                                     \
  INIT_IPC_NS(ipc_ns)      \
+ .pts_ns  = &init_pts_ns,     \
  .user_ns = &init_user_ns,    \
 }

Index: 2.6.25-rc5-mm1/include/linux/devpts_fs.h
===================================================================
--- 2.6.25-rc5-mm1.orig/include/linux/devpts_fs.h 2008-04-02 22:51:59.000000000 -0700
```

```
+++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-04-02 22:51:59.000000000 -0700
@@ -31,7 +31,7 @@ extern struct pts_namespace init_pts_ns;

 static inline struct pts_namespace *current_pts_ns(void)
 {
- return &init_pts_ns;
+ return current->nsproxy->pts_ns;
 }

 static inline struct pts_namespace *pts_ns_from_inode(struct inode *inode)
@@ -61,7 +61,8 @@ struct tty_struct *devpts_get_tty(struct
 /* unlink */
 void devpts_pty_kill(struct pts_namespace *pts_ns, int number);

-static inline void free_pts_ns(struct kref *ns_kref) { }
+extern struct pts_namespace *new_pts_ns(void);
+extern void free_pts_ns(struct kref *kref);

 static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
 {
@@ -75,6 +76,15 @@ static inline void put_pts_ns(struct pts
   kref_put(&ns->kref, free_pts_ns);
 }

+static inline struct pts_namespace *copy_pts_ns(u64 flags,
+                struct pts_namespace *old_ns)
+{
+       if (flags & CLONE_NEWPTS)
+                return new_pts_ns();
+       else
+                return get_pts_ns(old_ns);
+}
+
 #else

 /* Dummy stubs in the no-pty case */
@@ -90,6 +100,14 @@ static inline struct pts_namespace *get_
 }

 static inline void put_pts_ns(struct pts_namespace *ns) { }
+
+static inline struct pts_namespace *copy_pts_ns(u64 flags,
+                struct pts_namespace *old_ns)
+{
+       if (flags & CLONE_NEWPTS)
+                return ERR_PTR(-EINVAL);
+       return old_ns;
+}
```

```
 #endif


Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-04-02 22:51:59.000000000 -0700
+++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-04-02 22:51:59.000000000 -0700
@@ -27,6 +27,7 @@

 extern int pty_limit;   /* Config limit on Unix98 ptys */
 static DECLARE_MUTEX(allocated_ptys_lock);
+static struct file_system_type devpts_fs_type;

 static struct {
  int setuid;
@@ -119,6 +120,57 @@ static const struct super_operations dev
  .show_options = devpts_show_options,
 };

+struct pts_namespace *new_pts_ns(void)
+{
+ struct pts_namespace *ns;
+
+ ns = kmalloc(sizeof(*ns), GFP_KERNEL);
+ if (!ns)
+  return ERR_PTR(-ENOMEM);
+
+ kref_init(&ns->kref);
+
+ ns->mnt = kern_mount_data(&devpts_fs_type, ns);
+ if (IS_ERR(ns->mnt)) {
+  kfree(ns);
+  return ERR_PTR(PTR_ERR(ns->mnt));
+ }
+
+ idr_init(&ns->allocated_ptys);
+
+ printk(KERN_NOTICE "Created pts-ns 0x%p\n", ns);
+
+ return ns;
+}
+
+void free_pts_ns(struct kref *ns_kref)
+{
+ struct pts_namespace *ns;
+
+ ns = container_of(ns_kref, struct pts_namespace, kref);
+
```

```
+ /*
+  * Clear s_fs_info here rather than in ->kill_sb(), since the pts_ns
+  * is invalid real soon now, but the ->kill_sb() will not happen
+  * until the last mntput(). And if some one is accessing this
+  * devpts mount from an ancestor pts ns, we may not be holding
+  * the last reference to this mnt.
+  *
+  * After clearing the pts_ns is NULL here, any acceses from parent
+  * will fail.
+  */
+ rcu_assign_pointer(ns->mnt->mnt_sb->s_fs_info, NULL);
+ mntput(ns->mnt);
+
+ /*
+  * TODO:
+  *      idr_remove_all(&ns->allocated_ptys); introduced in 2.6.23
+  */
+ idr_destroy(&ns->allocated_ptys);
+ kfree(ns);
+
+ printk(KERN_NOTICE "Freed pts-ns 0x%p\n", ns);
+}

 static int devpts_mknod(struct inode *dir, struct dentry *dentry,
    int mode, dev_t rdev)
@@ -217,7 +269,16 @@ static int devpts_test_sb(struct super_b

 static int devpts_set_sb(struct super_block *sb, void *data)
 {
- sb->s_fs_info = data;
+ /*
+  * new_pts_ns() mounts the pts namespace and free_pts_ns()
+  * drops the reference to the mount. i.e the s_fs_info is
+  * cleared and vfsmnt is releasand _before_ pts_namespace
+  * is freed.
+  *
+  * So we don't need a reference to the pts_namespace here
+  * (Getting a reference here will also cause circular reference).
+  */
+ rcu_assign_pointer(sb->s_fs_info, data);
  return set_anon_super(sb, NULL);
 }

@@ -232,7 +293,7 @@ static int devpts_get_sb(struct file_sys
  if (flags & MS_KERNMOUNT)
   ns = data;
  else
-  ns = &init_pts_ns;
```

```
+  ns = current_pts_ns();

  /* hereafter we're very simlar to get_sb_nodev */
  sb = sget(fs_type, devpts_test_sb, devpts_set_sb, ns);
@@ -286,6 +347,13 @@ int devpts_new_index(struct pts_namespac
  int index;
  int idr_ret;

+ /*
+  * If pts_ns is NULL, this must be an access from an ancestor-pts-ns
+  * which happened just as this pts-ns was freed. Fail the access
+  * from parent-pts-ns.
+  */
+ if (!pts_ns)
+  return -EAGAIN;
 retry:
  if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
   return -ENOMEM;
@@ -311,6 +379,7 @@ retry:

 void devpts_kill_index(struct pts_namespace *pts_ns, int idx)
 {
+ BUG_ON(pts_ns == NULL);

  down(&allocated_ptys_lock);
  idr_remove(&pts_ns->allocated_ptys, idx);
@@ -330,6 +399,7 @@ int devpts_pty_new( struct pts_namespace
  /* 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);
+ BUG_ON(pts_ns == NULL);

  mnt = pts_ns->mnt;
  root = mnt->mnt_root;
@@ -370,6 +440,14 @@ struct tty_struct *devpts_get_tty(struct
  struct dentry *dentry;
  struct tty_struct *tty;

+ /*
+  * If pts_ns is NULL, this must be an access from an ancestor-pts-ns
+  * which happened just as this pts-ns was freed. Fail the access
+  * from parent-pts-ns.
+  */
+ if (!pts_ns)
+  return ERR_PTR(-EAGAIN);
+
  mnt = pts_ns->mnt;
```

```
  dentry = get_node(mnt->mnt_root, number);
@@ -391,6 +469,8 @@ void devpts_pty_kill(struct pts_namespac
  struct dentry *dentry;
  struct dentry *root;

+ BUG_ON(pts_ns == NULL);
+
  root = pts_ns->mnt->mnt_root;

  dentry = get_node(root, number);
Index: 2.6.25-rc5-mm1/kernel/fork.c
===================================================================
--- 2.6.25-rc5-mm1.orig/kernel/fork.c 2008-04-02 22:50:22.000000000 -0700
+++ 2.6.25-rc5-mm1/kernel/fork.c 2008-04-02 22:51:59.000000000 -0700
@@ -1713,7 +1713,7 @@ static long do_unshare(u64 unshare_flags
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
     CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
-    CLONE_NEWNET))
+    CLONE_NEWNET|CLONE_NEWPTS))
   goto bad_unshare_out;

  if ((err = unshare_thread(unshare_flags)))
Index: 2.6.25-rc5-mm1/kernel/nsproxy.c
===================================================================
--- 2.6.25-rc5-mm1.orig/kernel/nsproxy.c 2008-04-02 22:50:22.000000000 -0700
+++ 2.6.25-rc5-mm1/kernel/nsproxy.c 2008-04-02 22:51:59.000000000 -0700
@@ -21,6 +21,7 @@
 #include <linux/utsname.h>
 #include <linux/pid_namespace.h>
 #include <net/net_namespace.h>
+#include <linux/devpts_fs.h>
 #include <linux/ipc_namespace.h>

 static struct kmem_cache *nsproxy_cachep;
@@ -93,8 +94,17 @@ static struct nsproxy *create_new_namesp
   goto out_net;
  }

+ new_nsp->pts_ns = copy_pts_ns(flags, tsk->nsproxy->pts_ns);
+ if (IS_ERR(new_nsp->pts_ns)) {
+  err = PTR_ERR(new_nsp->pts_ns);
+  goto out_pts;
+ }
+
  return new_nsp;

+out_pts:
```

```
+ if (new_nsp->net_ns)
+  put_net(new_nsp->net_ns);
 out_net:
  if (new_nsp->user_ns)
   put_user_ns(new_nsp->user_ns);
@@ -131,7 +141,8 @@ int copy_namespaces(u64 flags, struct ta
  get_nsproxy(old_ns);

  if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET)))
+    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET |
+    CLONE_NEWPTS)))
   return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -170,6 +181,8 @@ void free_nsproxy(struct nsproxy *ns)
   put_pid_ns(ns->pid_ns);
  if (ns->user_ns)
   put_user_ns(ns->user_ns);
+ if (ns->pts_ns)
+  put_pts_ns(ns->pts_ns);
  put_net(ns->net_ns);
  kmem_cache_free(nsproxy_cachep, ns);
 }
@@ -184,7 +197,7 @@ int unshare_nsproxy_namespaces(u64 unsha
  int err = 0;

  if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-         CLONE_NEWUSER | CLONE_NEWNET)))
+         CLONE_NEWUSER | CLONE_NEWNET | CLONE_NEWPTS)))
   return 0;

  if (!capable(CAP_SYS_ADMIN))
```

Subject: Re: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode
Posted by serue on Fri, 04 Apr 2008 16:32:19 GMT
View Forum Message <> Reply to Message

Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode.
>

> The devpts interfaces currently operate on a specific pts namespace
> which they get from the 'current' task.
>
> With implementation of containers and cloning of PTS namespaces, we want
> to be able to access PTYs in a child-pts-ns from a parent-pts-ns. For
> instance we could bind-mount and pivot-root the child container on
> '/vserver/vserver1' and then access the "pts/0" of 'vserver1' using
>
> $ echo foo > /vserver/vserver1/dev/pts/0
>
> The task doing the above 'echo' could be in parent-pts-ns. So we find
> the 'pts-ns' of the above file from the inode representing the above
> file rather than from the 'current' task.
>
> Note that we need to find and hold a reference to the pts_ns to prevent
> the pts_ns from being freed while it is being accessed from 'outside'.
>
> This patch implements, 'pts_ns_from_inode()' which returns the pts_ns
> using 'inode->i_sb->s_fs_info'.
>
> Since, the 'inode' information is not visible inside devpts code itself,
> this patch modifies the tty driver code to determine the pts_ns and passes
> it into devpts.
>
> Changelog [v2]:
> [Serge Hallyn] Use rcu to access sb->s_fs_info.
>
> [Serge Hallyn] Simplify handling of ptmx and tty devices by expecting
>   user to create them in /dev/pts (see also devpts-mknod patch)
>
>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> drivers/char/pty.c | 13 +++++-
> drivers/char/tty_io.c | 93 ++++++++++++++++++++++++++++++++++++++-------
> fs/devpts/inode.c | 19 +++++---
> include/linux/devpts_fs.h | 38 +++++++++++++++--
> 4 files changed, 131 insertions(+), 32 deletions(-)
>
> Index: 2.6.25-rc5-mm1/include/linux/devpts_fs.h
> ===
> --- 2.6.25-rc5-mm1.orig/include/linux/devpts_fs.h 2008-04-02 22:42:08.000000000 -0700
> +++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-04-02 22:42:14.000000000 -0700
> @@ -17,6 +17,7 @@
> #include <linux/nsproxy.h>
> #include <linux/kref.h>
> #include <linux/idr.h>
> +#include <linux/fs.h>

>
> struct pts_namespace {
>   struct kref kref;
> @@ -26,12 +27,39 @@ struct pts_namespace {
>
> extern struct pts_namespace init_pts_ns;
>
> +#define DEVPTS_SUPER_MAGIC 0x1cd1
> +
> +static inline struct pts_namespace *current_pts_ns(void)
> +{
> + return &init_pts_ns;
> +}
> +
> +static inline struct pts_namespace *pts_ns_from_inode(struct inode *inode)
> +{
> + /*
> + * If this file exists on devpts, return the pts_ns from the
> + * devpts super-block. Otherwise just use the pts-ns of the
> + * calling task.
> + */
> + if(inode->i_sb->s_magic == DEVPTS_SUPER_MAGIC)
> + return rcu_dereference(inode->i_sb->s_fs_info);
> +
> + return current_pts_ns();
> +}
> +
> +
> #ifdef CONFIG_UNIX98_PTYS
> -int devpts_new_index(void);
> -void devpts_kill_index(int idx);
> -int devpts_pty_new(struct tty_struct *tty);   /* mknod in devpts */
> -struct tty_struct *devpts_get_tty(int number); /* get tty structure */
> -void devpts_pty_kill(int number);   /* unlink */
> +int devpts_new_index(struct pts_namespace *pts_ns);
> +void devpts_kill_index(struct pts_namespace *pts_ns, int idx);
> +
> +/* mknod in devpts */
> +int devpts_pty_new(struct pts_namespace *pts_ns, struct tty_struct *tty);
> +
> +/* get tty structure */
> +struct tty_struct *devpts_get_tty(struct pts_namespace *pts_ns, int number);
> +
> +/* unlink */
> +void devpts_pty_kill(struct pts_namespace *pts_ns, int number);
>
> static inline void free_pts_ns(struct kref *ns_kref) { }
>

> Index: 2.6.25-rc5-mm1/drivers/char/tty_io.c
> ===
> --- 2.6.25-rc5-mm1.orig/drivers/char/tty_io.c 2008-04-02 22:35:29.000000000 -0700
> +++ 2.6.25-rc5-mm1/drivers/char/tty_io.c 2008-04-02 22:42:14.000000000 -0700
> @@ -2064,8 +2064,8 @@ static void tty_line_name(struct tty_dri
> * relaxed for the (most common) case of reopening a tty.
> */
>
> -static int init_dev(struct tty_driver *driver, int idx,
> - struct tty_struct **ret_tty)
> +static int init_dev(struct tty_driver *driver, struct pts_namespace *pts_ns,
> + int idx, struct tty_struct **ret_tty)
> {
> struct tty_struct *tty, *o_tty;
> struct ktermios *tp, **tp_loc, *o_tp, **o_tp_loc;
> @@ -2074,7 +2074,11 @@ static int init_dev(struct tty_driver *d
>
> /* check whether we're reopening an existing tty */
> if (driver->flags & TTY_DRIVER_DEVPTS_MEM) {
> - tty = devpts_get_tty(idx);
> + tty = devpts_get_tty(pts_ns, idx);
> + if (IS_ERR(tty)) {
> + retval = PTR_ERR(tty);
> + goto end_init;
> + }
> /*
> * If we don't have a tty here on a slave open, it's because
> * the master already started the close process and there's
> @@ -2361,6 +2365,31 @@ static void release_tty(struct tty_struc
> }
>
> /*
> + * When opening /dev/tty and /dev/ptmx, use the pts-ns of the calling
> + * process. For any other pts device, use the pts-ns, in which the
> + * device was created. This latter case is needed when the pty is
> + * being accessed from a parent container.
> + *
> + * Eg: Suppose the user used bind-mount and pivot-root to mount a
> + * child- container's root on /vs/vs1. Then "/vs/vs1/dev/pts/0"
> + * in parent container and "/dev/pts/0" in child container would
> + * refer to the same device.
> + *
> + * When parent-container opens, "/vs/vs1/dev/pts/0" we find and
> + * grab/drop reference to child container's pts-ns (using @filp).
> + */
> +struct pts_namespace *pty_pts_ns(struct tty_driver *driver, struct inode *inode)
> +{
> + struct pts_namespace *pts_ns;

```
> +
> + pts_ns = NULL;
> + if (driver->flags & TTY_DRIVER_DEVPTS_MEM)
> +  pts_ns = pts_ns_from_inode(inode);
> +
> + return pts_ns;
> +}
> +
> +/*
>   * Even releasing the tty structures is a tricky business.. We have
>   * to be very careful that the structures are all released at the
>   * same time, as interrupts might otherwise get the wrong pointers.
> @@ -2376,10 +2405,12 @@ static void release_dev(struct file *fil
>   int idx;
>   char buf[64];
>   unsigned long flags;
> + struct pts_namespace *pts_ns;
> + struct inode *inode;
>
> + inode = filp->f_path.dentry->d_inode;
>   tty = (struct tty_struct *)filp->private_data;
> - if (tty_paranoia_check(tty, filp->f_path.dentry->d_inode,
> -       "release_dev"))
> + if (tty_paranoia_check(tty, inode, "release_dev"))
>    return;
>
>   check_tty_count(tty, "release_dev");
> @@ -2392,6 +2423,12 @@ static void release_dev(struct file *fil
>   devpts = (tty->driver->flags & TTY_DRIVER_DEVPTS_MEM) != 0;
>   o_tty = tty->link;
>
> + /*
> +  * We already have a reference to pts_ns here, so it cannot
> +  * be going away.
> +  */
> + pts_ns = pty_pts_ns(tty->driver, inode);
> +
>  #ifdef TTY_PARANOIA_CHECK
>   if (idx < 0 || idx >= tty->driver->num) {
>    printk(KERN_DEBUG "release_dev: bad idx when trying to "
> @@ -2569,6 +2606,10 @@ static void release_dev(struct file *fil
>
>   mutex_unlock(&tty_mutex);
>
> + /* drop the reference from ptmx_open/tty_open() */
> + if (devpts)
> +  put_pts_ns(pts_ns);
> +
```

```
>   /* check whether both sides are closing ... */
>   if (!tty_closing || (o_tty && !o_tty_closing))
>    return;
> @@ -2634,7 +2675,7 @@ static void release_dev(struct file *fil
>
>   /* Make this pty number available for reallocation */
>   if (devpts)
> -  devpts_kill_index(idx);
> +  devpts_kill_index(pts_ns, idx);
>  }
>
>  /**
> @@ -2666,6 +2707,7 @@ static int tty_open(struct inode *inode,
>   int index;
>   dev_t device = inode->i_rdev;
>   unsigned short saved_flags = filp->f_flags;
> + struct pts_namespace *pts_ns;
>
>   nonseekable_open(inode, filp);
>
> @@ -2715,10 +2757,27 @@ retry_open:
>    return -ENODEV;
>   }
>  got_driver:
> - retval = init_dev(driver, index, &tty);
> +
> + /*
> +  * What pts-ns do we want to use when opening "/dev/tty" ?
> +  * Sounds like current_pts_ns(), but what should happen
> +  * if parent pts ns does:
> +  *
> +  *  echo foo > /vs/vs1/dev/tty
```

You'll want to remove this comment, right? Your patch 4 solved
this problem?

-serge

```
> +  * (See Serge's setupvs1 script for the /vs/vs1...)
> +  */
> + rcu_read_lock();
> + pts_ns = pty_pts_ns(driver, inode);
> + get_pts_ns(pts_ns);
> + rcu_read_unlock();
> +
> + retval = init_dev(driver, pts_ns, index, &tty);
>   mutex_unlock(&tty_mutex);
> - if (retval)
```

```
> + if (retval) {
> +  put_pts_ns(pts_ns);
>    return retval;
> + }
>
>   filp->private_data = tty;
>   file_move(filp, &tty->tty_files);
> @@ -2790,16 +2849,22 @@ static int ptmx_open(struct inode *inode
>   struct tty_struct *tty;
>   int retval;
>   int index;
> + struct pts_namespace *pts_ns;
>
>   nonseekable_open(inode, filp);
>
> + rcu_read_lock();
> + pts_ns = pts_ns_from_inode(inode);
> + get_pts_ns(pts_ns);
> + rcu_read_unlock();
> +
>   /* find a device that is not in use. */
> - index = devpts_new_index();
> + retval = index = devpts_new_index(pts_ns);
>   if (index < 0)
> -  return index;
> +  goto drop_ns;
>
>   mutex_lock(&tty_mutex);
> - retval = init_dev(ptm_driver, index, &tty);
> + retval = init_dev(ptm_driver, pts_ns, index, &tty);
>   mutex_unlock(&tty_mutex);
>
>   if (retval)
> @@ -2809,7 +2874,7 @@ static int ptmx_open(struct inode *inode
>   filp->private_data = tty;
>   file_move(filp, &tty->tty_files);
>
> - retval = devpts_pty_new(tty->link);
> + retval = devpts_pty_new(pts_ns, tty->link);
>   if (retval)
>    goto out1;
>
> @@ -2821,7 +2886,9 @@ out1:
>   release_dev(filp);
>   return retval;
>  out:
> - devpts_kill_index(index);
> + devpts_kill_index(pts_ns, index);
```

> +drop_ns:
> + put_pts_ns(pts_ns);
> return retval;
> }
> #endif
> Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
> ===
> --- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-04-02 22:42:01.000000000 -0700
> +++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-04-02 22:42:14.000000000 -0700
> @@ -23,8 +23,6 @@
> #include <linux/fsnotify.h>
> #include <linux/seq_file.h>
>
> -#define DEVPTS_SUPER_MAGIC 0x1cd1
> -
> #define DEVPTS_DEFAULT_MODE 0600
>
> extern int pty_limit; /* Config limit on Unix98 ptys */
> @@ -283,11 +281,10 @@ static struct dentry *get_node(struct de
> return lookup_one_len(s, root, sprintf(s, "%d", num));
> }
>
> -int devpts_new_index(void)
> +int devpts_new_index(struct pts_namespace *pts_ns)
> {
> int index;
> int idr_ret;
> - struct pts_namespace *pts_ns = &init_pts_ns;
>
> retry:
> if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
> @@ -312,16 +309,15 @@ retry:
> return index;
> }
>
> -void devpts_kill_index(int idx)
> +void devpts_kill_index(struct pts_namespace *pts_ns, int idx)
> {
> - struct pts_namespace *pts_ns = &init_pts_ns;
>
> down(&allocated_ptys_lock);
> idr_remove(&pts_ns->allocated_ptys, idx);
> up(&allocated_ptys_lock);
> }
>
> -int devpts_pty_new(struct tty_struct *tty)
> +int devpts_pty_new(struct pts_namespace *pts_ns, struct tty_struct *tty)
> {

```
>   int number = tty->index; /* tty layer puts index from devpts_new_index() in here */
>   struct tty_driver *driver = tty->driver;
> @@ -330,7 +326,6 @@ int devpts_pty_new(struct tty_struct *tt
>   struct dentry *root;
>   struct vfsmount *mnt;
>   struct inode *inode;
> - struct pts_namespace *pts_ns = &init_pts_ns;
>
>   /* We're supposed to be given the slave end of a pty */
>   BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
> @@ -369,13 +364,13 @@ int devpts_pty_new(struct tty_struct *tt
>   return 0;
>  }
>
> -struct tty_struct *devpts_get_tty(int number)
> +struct tty_struct *devpts_get_tty(struct pts_namespace *pts_ns, int number)
>  {
>   struct vfsmount *mnt;
>   struct dentry *dentry;
>   struct tty_struct *tty;
>
> - mnt = init_pts_ns.mnt;
> + mnt = pts_ns->mnt;
>
>   dentry = get_node(mnt->mnt_root, number);
>
> @@ -391,12 +386,12 @@ struct tty_struct *devpts_get_tty(int nu
>   return tty;
>  }
>
> -void devpts_pty_kill(int number)
> +void devpts_pty_kill(struct pts_namespace *pts_ns, int number)
>  {
>   struct dentry *dentry;
>   struct dentry *root;
>
> - root = init_pts_ns.mnt->mnt_root;
> + root = pts_ns->mnt->mnt_root;
>
>   dentry = get_node(root, number);
>
> Index: 2.6.25-rc5-mm1/drivers/char/pty.c
> ===================================================================
> --- 2.6.25-rc5-mm1.orig/drivers/char/pty.c 2008-04-02 22:35:29.000000000 -0700
> +++ 2.6.25-rc5-mm1/drivers/char/pty.c 2008-04-02 22:42:14.000000000 -0700
> @@ -37,6 +37,9 @@ static struct tty_driver *pts_driver;
>
>  static void pty_close(struct tty_struct * tty, struct file * filp)
```

```
>  {
> + struct inode *inode;
> + struct pts_namespace *pts_ns;
> +
>   if (!tty)
>    return;
>   if (tty->driver->subtype == PTY_TYPE_MASTER) {
> @@ -58,8 +61,14 @@ static void pty_close(struct tty_struct
>   if (tty->driver->subtype == PTY_TYPE_MASTER) {
>    set_bit(TTY_OTHER_CLOSED, &tty->flags);
>  #ifdef CONFIG_UNIX98_PTYS
> -  if (tty->driver == ptm_driver)
> -   devpts_pty_kill(tty->index);
> +  if (tty->driver == ptm_driver) {
> +   inode = filp->f_path.dentry->d_inode;
> +   rcu_read_lock();
> +   pts_ns = pts_ns_from_inode(inode);
> +   rcu_read_unlock();
> +
> +   devpts_pty_kill(pts_ns, tty->index);
> +  }
>  #endif
>    tty_vhangup(tty->link);
>   }
```

_____

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

---

Subject: Re: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode
Posted by Sukadev Bhattiprolu on Fri, 04 Apr 2008 17:37:40 GMT
View Forum Message <> Reply to Message

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

```
| > +
| > + /*
| > +  * What pts-ns do we want to use when opening "/dev/tty" ?
| > +  * Sounds like current_pts_ns(), but what should happen
| > +  * if parent pts ns does:
| > +  *
| > +  *  echo foo > /vs/vs1/dev/tty
|
| You'll want to remove this comment, right?  Your patch 4 solved
| this problem?
```

Yes, Will remove while porting to rc8-mm1.

Should I go ahead and post as RFC to lkml ?

Sukadev

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

---

Subject: Re: [PATCH 6/7][v2]: Determine pts_ns from a pty's inode
Posted by serue on Fri, 04 Apr 2008 20:17:44 GMT
View Forum Message <> Reply to Message

Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):
> Serge E. Hallyn [serue@us.ibm.com] wrote:
> | > +
> | > + /*
> | > + * What pts-ns do we want to use when opening "/dev/tty" ?
> | > + * Sounds like current_pts_ns(), but what should happen
> | > + * if parent pts ns does:
> | > + *
> | > + * echo foo > /vs/vs1/dev/tty
> |
> | You'll want to remove this comment, right? Your patch 4 solved
> | this problem?
>
> Yes, Will remove while porting to rc8-mm1.
>
> Should I go ahead and post as RFC to lkml ?
>
> Sukadev

Well I haven't given it enough scrutiny to really do justice
to the change you've made. But the overall approach looks
right, and since I'll be on the road, don't hold off on my
account.

-serge

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

---