Subject: [RFC][PATCH 0/7] Clone PTS namespace
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 21:53:33 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. See

http://marc.info/?l=linux-kernel&m=120272411925609&w=2

- Needs some cleanup and more testing

- Ensure patchset is bisect-safe

Changelogs from earlier posts to Containers@.

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.

_____

Subject: [RFC][PATCH 1/7]: Propagate error code from devpts_pty_new
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 21:58:18 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 1/7]: Propagate error code from devpts_pty_new

Have ptmx_open() propagate any error code returned by devpts_pty_new()
(which returns either 0 or -ENOMEM anyway).

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 drivers/char/tty_io.c | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)

Index: 2.6.25-rc8-mm1/drivers/char/tty_io.c
==========
--- 2.6.25-rc8-mm1.orig/drivers/char/tty_io.c 2008-04-07 14:49:56.000000000 -0700
+++ 2.6.25-rc8-mm1/drivers/char/tty_io.c 2008-04-08 09:12:55.000000000 -0700
@@ -2835,8 +2835,8 @@ static int ptmx_open(struct inode *inode
  filp->private_data = tty;
  file_move(filp, &tty->tty_files);

```
- retval = -ENOMEM;
- if (devpts_pty_new(tty->link))
+ retval = devpts_pty_new(tty->link);
+ if (retval)
   goto out1;

  check_tty_count(tty, "tty_open");
```

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

---

Subject: [RFC][PATCH 2/7]: Factor out PTY index allocation
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 21:58:45 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 2/7]: Factor out PTY index allocation

Factor out the code used to allocate/free a pts index into new interfaces,
devpts_new_index() and devpts_kill_index(). This localizes the external
data structures used in managing the pts indices.

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>

```
 drivers/char/tty_io.c     |   40 ++++++----------------------------
 fs/devpts/inode.c         |   42 ++++++++++++++++++++++++++++++++++++++++++-
 include/linux/devpts_fs.h |    4 ++++
 3 files changed, 51 insertions(+), 35 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 20:04:07.000000000 -0700
+++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-03-24 20:04:26.000000000 -0700
@@ -17,6 +17,8 @@

 #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 */
```

```
@@ -24,6 +26,8 @@ void devpts_pty_kill(int number);   /* u
 #else

 /* Dummy stubs in the no-pty case */
+static inline int devpts_new_index(void) { return -EINVAL; }
+static inline void devpts_kill_index(int idx) { }
 static inline int devpts_pty_new(struct tty_struct *tty) { return -EINVAL; }
 static inline struct tty_struct *devpts_get_tty(int number) { return NULL; }
 static inline void devpts_pty_kill(int number) { }
Index: 2.6.25-rc5-mm1/drivers/char/tty_io.c
===================================================================
--- 2.6.25-rc5-mm1.orig/drivers/char/tty_io.c 2008-03-24 20:04:07.000000000 -0700
+++ 2.6.25-rc5-mm1/drivers/char/tty_io.c 2008-03-24 20:04:26.000000000 -0700
@@ -91,7 +91,6 @@
 #include <linux/module.h>
 #include <linux/smp_lock.h>
 #include <linux/device.h>
-#include <linux/idr.h>
 #include <linux/wait.h>
 #include <linux/bitops.h>
 #include <linux/delay.h>
@@ -137,9 +136,6 @@ EXPORT_SYMBOL(tty_mutex);

 #ifdef CONFIG_UNIX98_PTYS
 extern struct tty_driver *ptm_driver; /* Unix98 pty masters; for /dev/ptmx */
-extern int pty_limit;   /* Config limit on Unix98 ptys */
-static DEFINE_IDR(allocated_ptys);
-static DEFINE_MUTEX(allocated_ptys_lock);
 static int ptmx_open(struct inode *, struct file *);
 #endif

@@ -2636,15 +2632,9 @@ static void release_dev(struct file *fil
   */
  release_tty(tty, idx);

-#ifdef CONFIG_UNIX98_PTYS
  /* Make this pty number available for reallocation */
- if (devpts) {
-  mutex_lock(&allocated_ptys_lock);
-  idr_remove(&allocated_ptys, idx);
-  mutex_unlock(&allocated_ptys_lock);
- }
-#endif
-
+ if (devpts)
+  devpts_kill_index(idx);
 }
```

```
 /**
@@ -2800,29 +2790,13 @@ static int ptmx_open(struct inode *inode
  struct tty_struct *tty;
  int retval;
  int index;
- int idr_ret;

  nonseekable_open(inode, filp);

  /* find a device that is not in use. */
- mutex_lock(&allocated_ptys_lock);
- if (!idr_pre_get(&allocated_ptys, GFP_KERNEL)) {
-  mutex_unlock(&allocated_ptys_lock);
-  return -ENOMEM;
- }
- idr_ret = idr_get_new(&allocated_ptys, NULL, &index);
- if (idr_ret < 0) {
-  mutex_unlock(&allocated_ptys_lock);
-  if (idr_ret == -EAGAIN)
-   return -ENOMEM;
-  return -EIO;
- }
- if (index >= pty_limit) {
-  idr_remove(&allocated_ptys, index);
-  mutex_unlock(&allocated_ptys_lock);
-  return -EIO;
- }
- mutex_unlock(&allocated_ptys_lock);
+ index = devpts_new_index();
+ if (index < 0)
+  return index;

  mutex_lock(&tty_mutex);
  retval = init_dev(ptm_driver, index, &tty);
@@ -2847,9 +2821,7 @@ out1:
  release_dev(filp);
  return retval;
 out:
- mutex_lock(&allocated_ptys_lock);
- idr_remove(&allocated_ptys, index);
- mutex_unlock(&allocated_ptys_lock);
+ devpts_kill_index(index);
  return retval;
 }
 #endif
Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-03-24 20:04:07.000000000 -0700
```

```
+++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-03-24 20:04:26.000000000 -0700
@@ -17,6 +17,7 @@
 #include <linux/namei.h>
 #include <linux/mount.h>
 #include <linux/tty.h>
+#include <linux/idr.h>
 #include <linux/devpts_fs.h>
 #include <linux/parser.h>
 #include <linux/fsnotify.h>
@@ -26,6 +27,10 @@

 #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;

@@ -171,9 +176,44 @@ static struct dentry *get_node(int num)
  return lookup_one_len(s, root, sprintf(s, "%d", num));
 }

+int devpts_new_index(void)
+{
+ int index;
+ int idr_ret;
+
+retry:
+ if (!idr_pre_get(&allocated_ptys, GFP_KERNEL)) {
+  return -ENOMEM;
+ }
+
+ down(&allocated_ptys_lock);
+ idr_ret = idr_get_new(&allocated_ptys, NULL, &index);
+ if (idr_ret < 0) {
+  up(&allocated_ptys_lock);
+  if (idr_ret == -EAGAIN)
+   goto retry;
+  return -EIO;
+ }
+
+ if (index >= pty_limit) {
+  idr_remove(&allocated_ptys, index);
+  up(&allocated_ptys_lock);
+  return -EIO;
+ }
```

```
+ up(&allocated_ptys_lock);
+ return index;
+}
+
+void devpts_kill_index(int idx)
+{
+ down(&allocated_ptys_lock);
+ idr_remove(&allocated_ptys, idx);
+ up(&allocated_ptys_lock);
+}
+
 int devpts_pty_new(struct tty_struct *tty)
 {
- int number = tty->index;
+ int number = tty->index; /* tty layer puts index from devpts_new_index() in here */
  struct tty_driver *driver = tty->driver;
  dev_t device = MKDEV(driver->major, driver->minor_start+number);
  struct dentry *dentry;
```

Subject: [RFC][PATCH 3/7]: Enable multiple mounts of /dev/pts
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 21:59:09 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject:[RFC][PATCH 3/7]: 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: [RFC][PATCH 4/7]: Allow mknod of ptmx and tty in devpts
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 21:59:32 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 4/7]: Allow mknod of ptmx and tty in devpts

We want to allow administrators to access PTYs in descendant pts-namespaces,
for instance "echo foo > /vserver/vserver1/dev/pts/0". To enable such access
we must hold a reference to the pts-ns in which the device (ptmx or slave pty)
exists.

Note that we cannot use the pts-ns of the 'current' process since that pts-ns
could be different from the pts-ns in which the PTY device was created. So
we find the pts-ns from the inode of the PTY (inode->i_sb->s_fs_info).

While this would work for the slave PTY devices like /dev/pts/0, it would
not work for either the master PTY device (/dev/ptmx) or controlling terminal
(/dev/tty).

To uniformly handle the master, slave and controlling ttys, we allow creation
of 'ptmx' and 'tty' devices in /dev/pts. When creating containers, the
administrator can then:

In init-pts-ns:

```
 $ mknod /dev/pts/ptmx c 5 2
 $ mknod /dev/pts/tty c 5 0
 $ rm /dev/ptmx /dev/tty
 $ ln -s /dev/pts/ptmx /dev/ptmx
 $ ln -s /dev/pts/tty /dev/tty
```

In child-pts-ns:

```
 $ umount /dev/pts
 $ mount -t devpts lxcpts /dev/pts
 $ mknod /dev/pts/ptmx c 5 2
 $ mknod /dev/pts/tty c 5 0
```

With this, even if the 'ptmx' is accessed from parent pts-ns we still find
and hold the pts-ns in which 'ptmx' actually belongs.

This patch merely allows creation of /dev/pts/ptmx and /dev/pts/tty. Follow-on
patches will enable cloning the pts namespace and using the pts-ns from
the inode.

TODO:
 - Ability to unlink the /dev/pts/ptmx and /dev/pts/tty nodes.

Note:
 - If /dev/ptmx is a symlink to /vserver/vserver1/dev/pts/ptmx,
   open("/dev/ptmx") in init-pts-ns will create a PTY in 'vserver1' !

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
---
 fs/devpts/inode.c |   55 ++++++++++++++++++++++++++++++++++++++++++++++++++++---
 1 file changed, 51 insertions(+), 4 deletions(-)

Index: 2.6.25-rc8-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc8-mm1.orig/fs/devpts/inode.c 2008-04-08 09:18:23.000000000 -0700
+++ 2.6.25-rc8-mm1/fs/devpts/inode.c 2008-04-08 13:35:43.000000000 -0700
@@ -58,7 +58,6 @@ struct pts_namespace init_pts_ns = {
  .mnt = NULL,
 };

-
 static int devpts_remount(struct super_block *sb, int *flags, char *data)
 {
  char *p;
@@ -122,6 +121,54 @@ static const struct super_operations dev
  .show_options = devpts_show_options,
 };

+
+static int devpts_mknod(struct inode *dir, struct dentry *dentry,
+   int mode, dev_t rdev)
+{
+ int inum;
+ struct inode *inode;
+ struct super_block *sb = dir->i_sb;
+
+ if (dentry->d_inode)
+  return -EEXIST;
+
+ if (!S_ISCHR(mode))
+  return -EPERM;
+
+ if (rdev == MKDEV(TTYAUX_MAJOR, 0))
+  inum = 2;
+ else if (rdev == MKDEV(TTYAUX_MAJOR, 2))
+  inum = 3;
+ else
+  return -EPERM;
+
+ inode = new_inode(sb);
+ if (!inode)
+  return -ENOMEM;
+
+ inode->i_ino = inum;
+ inode->i_uid = inode->i_gid = 0;
+ inode->i_blocks = 0;
```

```
+ inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
+
+ init_special_inode(inode, mode, rdev);
+
+ d_instantiate(dentry, inode);
+ /*
+  * Get a reference to the dentry so the device-nodes persist
+  * even when there are no active references to them. We use
+  * kill_litter_super() to remove this entry when unmounting
+  * devpts.
+  */
+ dget(dentry);
+ return 0;
+}
+
+const struct inode_operations devpts_dir_inode_operations = {
+ .lookup         = simple_lookup,
+ .mknod  = devpts_mknod,
+};
+
 static int
 devpts_fill_super(struct super_block *s, void *data, int silent)
 {
@@ -141,7 +188,7 @@ devpts_fill_super(struct super_block *s,
  inode->i_blocks = 0;
  inode->i_uid = inode->i_gid = 0;
  inode->i_mode = S_IFDIR | S_IRUGO | S_IXUGO | S_IWUSR;
- inode->i_op = &simple_dir_inode_operations;
+ inode->i_op = &devpts_dir_inode_operations;
  inode->i_fop = &simple_dir_operations;
  inode->i_nlink = 2;

@@ -214,7 +261,7 @@ static int devpts_get_sb(struct file_sys
 static void devpts_kill_sb(struct super_block *sb)
 {
  sb->s_fs_info = NULL;
- kill_anon_super(sb);
+ kill_litter_super(sb);
 }

 static struct file_system_type devpts_fs_type = {
@@ -303,7 +350,7 @@ int devpts_pty_new(struct tty_struct *tt
  if (!inode)
   return -ENOMEM;

- inode->i_ino = number+2;
+ inode->i_ino = number+4;
  inode->i_uid = config.setuid ? config.uid : current->fsuid;
```

```
  inode->i_gid = config.setgid ? config.gid : current->fsgid;
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
```

_____

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

---

Subject: [RFC][PATCH 5/7]: Implement get_pts_ns() and put_pts_ns()
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 22:00:00 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 5/7]: Implement get_pts_ns() and put_pts_ns()

Implement get_pts_ns() and put_pts_ns() interfaces.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 include/linux/devpts_fs.h | 21 ++++++++++++++++++++-
 1 file changed, 20 insertions(+), 1 deletion(-)

Index: 2.6.25-rc8-mm1/include/linux/devpts_fs.h
```
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/devpts_fs.h 2008-04-08 09:18:23.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/devpts_fs.h 2008-04-08 13:36:31.000000000 -0700
@@ -27,13 +27,26 @@ struct pts_namespace {
 extern struct pts_namespace init_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 */

+static inline void free_pts_ns(struct kref *ns_kref) { }
+
+static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
+{
+ if (ns && (ns != &init_pts_ns))
+  kref_get(&ns->kref);
+ return ns;
+}
+static inline void put_pts_ns(struct pts_namespace *ns)
+{
+ if (ns && (ns != &init_pts_ns))
```

```
+  kref_put(&ns->kref, free_pts_ns);
+}
+
 #else

 /* Dummy stubs in the no-pty case */
@@ -43,6 +56,12 @@ static inline int devpts_pty_new(struct
 static inline struct tty_struct *devpts_get_tty(int number) { return NULL; }
 static inline void devpts_pty_kill(int number) { }

+static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
+{
+ return &init_pts_ns;
+}
+
+static inline void put_pts_ns(struct pts_namespace *ns) { }
 #endif
```

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

---

Subject: [RFC][PATCH 6/7]: Determine pts_ns from a pty's inode
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 22:00:26 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 6/7]: 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 device
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 | 96 ++++++++++++++++++++++++++++++++++++++-------
 fs/devpts/inode.c | 19 +++++---
 include/linux/devpts_fs.h | 38 +++++++++++++---
 4 files changed, 134 insertions(+), 32 deletions(-)

Index: 2.6.25-rc8-mm1/include/linux/devpts_fs.h
===
--- 2.6.25-rc8-mm1.orig/include/linux/devpts_fs.h 2008-04-08 13:36:31.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/devpts_fs.h 2008-04-08 13:38:08.000000000 -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-rc8-mm1/drivers/char/tty_io.c
===================================================================
--- 2.6.25-rc8-mm1.orig/drivers/char/tty_io.c 2008-04-08 09:15:56.000000000 -0700
+++ 2.6.25-rc8-mm1/drivers/char/tty_io.c 2008-04-08 14:25:11.000000000 -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 +2365,21 @@ static void release_tty(struct tty_struc
 }

 /*
+ * If the inode belongs to a device in devpts fs, return the pts-namespace
+ * associated with the device. Return NULL otherwise.
+ */
+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 +2395,12 @@ 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 +2413,12 @@ 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 +2596,10 @@ 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 +2665,7 @@ 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 +2697,7 @@ 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 +2747,31 @@ retry_open:
   return -ENODEV;
  }
 got_driver:
- retval = init_dev(driver, index, &tty);
+
```

```
+ /*
+  * If this is a pty device, we maybe accessing this device from
+  * an ancestor pts-namespace. Find the pts-namespace from the
+  * device's inode and grab a reference.
+  *
+  * If pts-namespace is NULL then:
+  *  - either this is not a PTY device or
+  *  - this open is from an ancestor-pts-ns and the pts-ns has
+  *    just been freed.
+  *
+  * If the pts-namespace is NULL for a PTY device (i.e pts-ns has
+  * been freed), init_dev() will fail the open()).
+  */
+ 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 +2843,31 @@ static int ptmx_open(struct inode *inode
  struct tty_struct *tty;
  int retval;
  int index;
+ struct pts_namespace *pts_ns;

  nonseekable_open(inode, filp);

+ /*
+  * We maybe accessing this device from an ancestor pts-namespace.
+  * Find the pts-namespace from the device's inode and grab a
+  * reference.
+  *
+  * If pts-namespace is NULL, this open is from an ancestor-pts-ns
+  * and the pts-ns has just been freed and devpts_new_index()
+  * below will fail the open().
+  */
+ 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 +2877,7 @@ 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 +2889,9 @@ 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-rc8-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc8-mm1.orig/fs/devpts/inode.c 2008-04-08 13:35:43.000000000 -0700
+++ 2.6.25-rc8-mm1/fs/devpts/inode.c 2008-04-08 13:38:08.000000000 -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-rc8-mm1/drivers/char/pty.c
===================================================================
--- 2.6.25-rc8-mm1.orig/drivers/char/pty.c 2008-04-08 09:12:54.000000000 -0700
+++ 2.6.25-rc8-mm1/drivers/char/pty.c 2008-04-08 13:37:16.000000000 -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: [RFC][PATCH 7/7]: Enable cloning PTY namespaces
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 22:00:54 GMT
View Forum Message <> Reply to Message

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

Enable cloning PTY namespaces.

Note:
 We are out of clone_flags! This patch depends on Cedric Le Goater's
 clone64() patchset.

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-rc8-mm1/include/linux/sched.h
```
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/sched.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/sched.h 2008-04-08 14:27:41.000000000 -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-rc8-mm1/include/linux/nsproxy.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/nsproxy.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/nsproxy.h 2008-04-08 14:27:41.000000000 -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-rc8-mm1/include/linux/init_task.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/init_task.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/init_task.h 2008-04-08 14:27:41.000000000 -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-rc8-mm1/include/linux/devpts_fs.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/devpts_fs.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/devpts_fs.h 2008-04-08 14:27:41.000000000 -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-rc8-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc8-mm1.orig/fs/devpts/inode.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/fs/devpts/inode.c 2008-04-08 14:33:04.000000000 -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 released _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-rc8-mm1/kernel/fork.c
===================================================================
--- 2.6.25-rc8-mm1.orig/kernel/fork.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/kernel/fork.c 2008-04-08 14:27:41.000000000 -0700
@@ -1714,7 +1714,7 @@ 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-rc8-mm1/kernel/nsproxy.c
===================================================================
--- 2.6.25-rc8-mm1.orig/kernel/nsproxy.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/kernel/nsproxy.c 2008-04-08 14:27:41.000000000 -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: [RFC][PATCH 0/7] Clone PTS namespace
Posted by hpa on Wed, 09 Apr 2008 00:53:31 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> 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.
>

Why do we "need" this? There isn't a fundamental need for this to be a
dense numberspace (in fact, there are substantial reasons why it's a bad

idea; the only reason the namespace is dense at the moment is because of
the hideously bad handing of utmp in glibc.) Other than indicies, this
seems to be a more special case of device isolation across namespaces,
would that be a more useful problem to solve across the board?

 hpa

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by Sukadev Bhattiprolu on Wed, 09 Apr 2008 16:23:53 GMT
View Forum Message <> Reply to Message

H. Peter Anvin [hpa@zytor.com] wrote:
> sukadev@us.ibm.com wrote:
>> 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.
>
> Why do we "need" this? There isn't a fundamental need for this to be a
> dense numberspace (in fact, there are substantial reasons why it's a bad
> idea; the only reason the namespace is dense at the moment is because of
> the hideously bad handing of utmp in glibc.) Other than indicies, this
> seems to be a more special case of device isolation across namespaces,
> would that be a more useful problem to solve across the board?

We want to provide isolation between containers, meaning PTYs in container
C1 should not be accessible to processes in C2 (unless C2 is an ancestor).

The other reason for this in the longer term is for checkpoint/restart.
When restarting an application we want to make sure that the PTY indices
it was using is available and isolated.

We started out with isolating just the indices but added the special-case
handling for granting the host visibility into a child-container.

A complete device-namespace could solve this, but IIUC, is being planned
in the longer term. We are hoping this would provide the isolation in the
near-term without being too intrusive or impeding the implementation of
the device namespace.

Sukadev

_____

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by hpa on Wed, 09 Apr 2008 18:01:16 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> We want to provide isolation between containers, meaning PTYs in container
> C1 should not be accessible to processes in C2 (unless C2 is an ancestor).

Yes, I certainly can understand the desire for isolation. That wasn't
what my question was about.

> The other reason for this in the longer term is for checkpoint/restart.
> When restarting an application we want to make sure that the PTY indices
> it was using is available and isolated.

OK, this would be the motivation for index isolation.

> A complete device-namespace could solve this, but IIUC, is being planned
> in the longer term. We are hoping this would provide the isolation in the
> near-term without being too intrusive or impeding the implementation of
> the device namespace.

I'm just worried about the accumulation of what feels like ad hoc
namespaces, causing a very large combination matrix, a lot of which
don't make sense.

-hpa

_____

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serge on Wed, 09 Apr 2008 19:16:34 GMT
View Forum Message <> Reply to Message

Quoting H. Peter Anvin (hpa@zytor.com):
> sukadev@us.ibm.com wrote:

>> We want to provide isolation between containers, meaning PTYs in container
>> C1 should not be accessible to processes in C2 (unless C2 is an ancestor).
>
> Yes, I certainly can understand the desire for isolation. That wasn't what
> my question was about.
>
>> The other reason for this in the longer term is for checkpoint/restart.
>> When restarting an application we want to make sure that the PTY indices
>> it was using is available and isolated.
>
> OK, this would be the motivation for index isolation.
>
>> A complete device-namespace could solve this, but IIUC, is being planned
>> in the longer term. We are hoping this would provide the isolation in the
>> near-term without being too intrusive or impeding the implementation of
>> the device namespace.
>
> I'm just worried about the accumulation of what feels like ad hoc
> namespaces, causing a very large combination matrix, a lot of which don't
> make sense.

Hmm, if we were to just call this CLONE_NEWDEV, would that (a) make
sense and (b) suitably address your (certainly valid) concern?

Basically for now CLONE_NEWDEV wouldn't yet be fully implemented, only
unsharing unix98 ptys...

-serge

_____

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by ebiederm on Wed, 09 Apr 2008 22:15:13 GMT
View Forum Message <> Reply to Message

On Tue, 2008-04-08 at 17:53 -0700, H. Peter Anvin wrote:
> sukadev@us.ibm.com wrote:
> > 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.
> >

>
> Why do we "need" this? There isn't a fundamental need for this to be a
> dense numberspace (in fact, there are substantial reasons why it's a bad
> idea; the only reason the namespace is dense at the moment is because of
> the hideously bad handing of utmp in glibc.) Other than indicies, this
> seems to be a more special case of device isolation across namespaces,
> would that be a more useful problem to solve across the board?

In short application migration. When you move a running applicaiton
from one machine to another you want to be able to keep the same pseudo
devices.

The isolation that you have noticed is also an important application and
like the rest of the namespaces if we can solve the duplicate identifier
problem needed to restore checkpoints we also largely solve the
isolation problem.

This problem is much larger then ptys. ptys are the really in your face
aspect of it. There are a more pseudo devices in the kernel and it is
the device number to device mapping that we are abstracting. So this
really should be done as a device namespace not a pty namespace.

I would be happy if the first version of the device namespace could not
map anything but pty's (assuming an incremental implementation path). I
really don't think we should do a special case for each kind of device.

Oh and just skimming the patch summary I'm pretty certain this
implementation breaks /sys/class/tty/ptyXX/uevent. Which is another
reason why it would be good to have a single device namespace. So we
only to capture one more namespace and figure out how to deal with it
when mounting sysfs.

Eric

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by hpa on Wed, 09 Apr 2008 22:38:08 GMT
View Forum Message <> Reply to Message

serge@hallyn.com wrote:
>> I'm just worried about the accumulation of what feels like ad hoc
>> namespaces, causing a very large combination matrix, a lot of which don't
>> make sense.

>
> Hmm, if we were to just call this CLONE_NEWDEV, would that (a) make
> sense and (b) suitably address your (certainly valid) concern?
>
> Basically for now CLONE_NEWDEV wouldn't yet be fully implemented, only
> unsharing unix98 ptys...

That would make sense to me. Also see Eric's note about uevent,
however; and there are probably other issues like it.

 -hpa

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Thu, 10 Apr 2008 01:59:55 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> On Tue, 2008-04-08 at 17:53 -0700, H. Peter Anvin wrote:
> > sukadev@us.ibm.com wrote:
> > > 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.
> > >
> >
> > Why do we "need" this? There isn't a fundamental need for this to be a
> > dense numberspace (in fact, there are substantial reasons why it's a bad
> > idea; the only reason the namespace is dense at the moment is because of
> > the hideously bad handing of utmp in glibc.) Other than indicies, this
> > seems to be a more special case of device isolation across namespaces,
> > would that be a more useful problem to solve across the board?
>
> In short application migration. When you move a running applicaiton
> from one machine to another you want to be able to keep the same pseudo
> devices.
>
> The isolation that you have noticed is also an important application and
> like the rest of the namespaces if we can solve the duplicate identifier
> problem needed to restore checkpoints we also largely solve the
> isolation problem.

>
> This problem is much larger then ptys. ptys are the really in your face
> aspect of it. There are a more pseudo devices in the kernel and it is
> the device number to device mapping that we are abstracting. So this
> really should be done as a device namespace not a pty namespace.
>
> I would be happy if the first version of the device namespace could not
> map anything but pty's (assuming an incremental implementation path). I
> really don't think we should do a special case for each kind of device.

Sounds like we're all agreed on this and just doing
s/CLONE_NEWPTS/CLONE_NEWDEV/ on the current patchset suffices for now.
But,

> Oh and just skimming the patch summary I'm pretty certain this
> implementation breaks /sys/class/tty/ptyXX/uevent. Which is another
> reason why it would be good to have a single device namespace. So we
> only to capture one more namespace and figure out how to deal with it
> when mounting sysfs.

Feh, so of course sysfs would have the most interactions for a device
namespace, but now we have pty, network, and user namespace all needing
some sort of sysfs solution. For a quickfix for
CONFIG_USER_SCHED+CONFIG_USER_NS, I just moved /sys/kernel/uids/<uid>
to /sys/kernel/uids/<userns_address>/<uid>. But what would be a *good*
general solution?

ln -s /sys /proc/self/sys?

-serge

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by ebiederm on Thu, 10 Apr 2008 07:36:20 GMT
View Forum Message <> Reply to Message

On Wed, 2008-04-09 at 20:59 -0500, Serge E. Hallyn wrote:

> Feh, so of course sysfs would have the most interactions for a device
> namespace, but now we have pty, network, and user namespace all needing
> some sort of sysfs solution. For a quickfix for
> CONFIG_USER_SCHED+CONFIG_USER_NS, I just moved /sys/kernel/uids/<uid>
> to /sys/kernel/uids/<userns_address>/<uid>. But what would be a *good*
> general solution?

>
> ln -s /sys /proc/self/sys?

LOL

In /proc I prefer the /proc/self approach because we can do it and it is just much easier to setup and use. (Plus we have weird problems if we try and capture more then the pid namespace).

For other filesystems the only really viable option is to capture namespaces at mount time, as we are doing for devpts and proc with respect to the pid namespace.

For the network namespace where it is very much more then a single directory with symlinks from physical devices pointing at logical network interfaces.

My last effort in that area was ok'd by Tejun lightly tested by a few others and misplaced by gregkh so I don't think we have a real problem with resurrecting those patches cleaning them up a bit and merging them.

The biggest gotcha with sysfs is that the VFS locking for the dcache is in the wrong order for distributed filesystems, where we would like to make the change atomically on the server and them make the change in the local cache. Or in this case the sysfs internal data structures. The truly nasty case is supporting rename (as sysfs does) as the VFS is not at all happy if you just punt and shoot down the dentries and instantiate new ones.

I'm hoping to be able to get back at this in the week or so as things settle down from my move. My last patches should be in my proof of concept network namespace tree, if they don't show up elsewhere.

It isn't my perception that we have a design problem rather, we just need to move an important piece of code in a subtle and moderately uninteresting direction for it's primary maintainer.

Further what I did for the network namespace should easily handle the uid/gid namespace and should be a good starting place for a general device namespace.

Eric

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Thu, 10 Apr 2008 16:44:36 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> On Wed, 2008-04-09 at 20:59 -0500, Serge E. Hallyn wrote:
>
> > Feh, so of course sysfs would have the most interactions for a device
> > namespace, but now we have pty, network, and user namespace all needing
> > some sort of sysfs solution. For a quickfix for
> > CONFIG_USER_SCHED+CONFIG_USER_NS, I just moved /sys/kernel/uids/<uid>
> > to /sys/kernel/uids/<userns_address>/<uid>. But what would be a *good*
> > general solution?
> >
> > ln -s /sys /proc/self/sys?
>
> LOL
>
> In /proc I prefer the /proc/self approach because we can do it and it is
> just much easier to setup and use. (Plus we have weird problems if we
> try and capture more then the pid namespace).
>
> For other filesystems the only really viable option is to capture
> namespaces at mount time, as we are doing for devpts and proc with
> respect to the pid namespace.
>
> For the network namespace where it is very much more then a single
> directory with symlinks from physical devices pointing at logical
> network interfaces.
>
> My last effort in that area was ok'd by Tejun lightly tested by a few
> others and misplaced by gregkh so I don't think we have a real problem
> with resurrecting those patches cleaning them up a bit and merging them.
>
> The biggest gotcha with sysfs is that the VFS locking for the dcache
> is in the wrong order for distributed filesystems, where we would like
> to make the change atomically on the server and them make the change in
> the local cache. Or in this case the sysfs internal data structures.
> The truly nasty case is supporting rename (as sysfs does) as the VFS
> is not at all happy if you just punt and shoot down the dentries and
> instantiate new ones.
>
> I'm hoping to be able to get back at this in the week or so as things
> settle down from my move. My last patches should be in my proof of
> concept network namespace tree, if they don't show up elsewhere.
>
> It isn't my perception that we have a design problem rather, we just
> need to move an important piece of code in a subtle and moderately
> uninteresting direction for it's primary maintainer.

>
> Further what I did for the network namespace should easily handle the
> uid/gid namespace and should be a good starting place for a general
> device namespace.

Agreed. What's the git url and which branch do i use for your proof
of concept tree? I'll do the userns patch on top of that. I assume
Suka will do the same for ptys?

thanks,
-serge

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by Sukadev Bhattiprolu on Thu, 10 Apr 2008 20:58:00 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn [serue@us.ibm.com] wrote:
| >
| > Further what I did for the network namespace should easily handle the
| > uid/gid namespace and should be a good starting place for a general
| > device namespace.
|
| Agreed. What's the git url and which branch do i use for your proof
| of concept tree? I'll do the userns patch on top of that. I assume
| Suka will do the same for ptys?
|

Sure.

BTW, can we push the following 3 helper patches in the set. I believe
they will be required to support multiple pts namespaces, even if
the actual way we do it is not final yet.

    [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

Sukadev

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Tue, 22 Apr 2008 14:25:39 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> On Wed, 2008-04-09 at 20:59 -0500, Serge E. Hallyn wrote:
>
> > Feh, so of course sysfs would have the most interactions for a device
> > namespace, but now we have pty, network, and user namespace all needing
> > some sort of sysfs solution. For a quickfix for
> > CONFIG_USER_SCHED+CONFIG_USER_NS, I just moved /sys/kernel/uids/<uid>
> > to /sys/kernel/uids/<userns_address>/<uid>. But what would be a *good*
> > general solution?
> >
> > ln -s /sys /proc/self/sys?
>
> LOL
>
> In /proc I prefer the /proc/self approach because we can do it and it is
> just much easier to setup and use. (Plus we have weird problems if we
> try and capture more then the pid namespace).
>
> For other filesystems the only really viable option is to capture
> namespaces at mount time, as we are doing for devpts and proc with
> respect to the pid namespace.
>
> For the network namespace where it is very much more then a single
> directory with symlinks from physical devices pointing at logical
> network interfaces.
>
> My last effort in that area was ok'd by Tejun lightly tested by a few
> others and misplaced by gregkh so I don't think we have a real problem
> with resurrecting those patches cleaning them up a bit and merging them.
>
> The biggest gotcha with sysfs is that the VFS locking for the dcache
> is in the wrong order for distributed filesystems, where we would like
> to make the change atomically on the server and them make the change in
> the local cache. Or in this case the sysfs internal data structures.
> The truly nasty case is supporting rename (as sysfs does) as the VFS
> is not at all happy if you just punt and shoot down the dentries and
> instantiate new ones.
>
> I'm hoping to be able to get back at this in the week or so as things
> settle down from my move. My last patches should be in my proof of
> concept network namespace tree, if they don't show up elsewhere.

Is that the tree I'd get from

git-fetch

git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
master:ebieder.master

? So I'd add a user_ns to the struct sysfs_tag_info?

If so I'll give it a whirl.

thanks,
-serge

> It isn't my perception that we have a design problem rather, we just
> need to move an important piece of code in a subtle and moderately
> uninteresting direction for it's primary maintainer.
>
> Further what I did for the network namespace should easily handle the
> uid/gid namespace and should be a good starting place for a general
> device namespace.
>
> Eric
>

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by ebiederm on Tue, 22 Apr 2008 18:53:07 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:
>>
>> I'm hoping to be able to get back at this in the week or so as things
>> settle down from my move. My last patches should be in my proof of
>> concept network namespace tree, if they don't show up elsewhere.
>
> Is that the tree I'd get from
>
> git-fetch
> git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
> master:ebieder.master

Yes.

> ? So I'd add a user_ns to the struct sysfs_tag_info?
>
> If so I'll give it a whirl.

Sounds good. My apologies I keep being almost on the verge
of getting someplace.

Eric

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Wed, 23 Apr 2008 14:36:10 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >>
> >> I'm hoping to be able to get back at this in the week or so as things
> >> settle down from my move. My last patches should be in my proof of
> >> concept network namespace tree, if they don't show up elsewhere.
> >
> > Is that the tree I'd get from
> >
> > git-fetch
> > git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
> > master:ebieder.master
>
> Yes.
>
> > ? So I'd add a user_ns to the struct sysfs_tag_info?
> >
> > If so I'll give it a whirl.
>
> Sounds good. My apologies I keep being almost on the verge
> of getting someplace.

Ok I've got the sysfs relevant patches ported to 2.6.25, and am looking
at how to extend it to handle /sys/kernel/uids. You have tagging tied
intimately to struct class. So the question is should I generalize the
taggint to deal with kobjects instead, or create a struct class user
and make /sys/kernel/uids a symlink to /sys/class/user/uids?

Opinions?

thanks,
-serge

PS: If you want me to post the patchset before handling userns I can do

that, otherwise I was just going to wait until I'm done with userns.

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Wed, 23 Apr 2008 17:57:49 GMT
View Forum Message <> Reply to Message

Quoting Serge E. Hallyn (serue@us.ibm.com):
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > >>
> > >> I'm hoping to be able to get back at this in the week or so as things
> > >> settle down from my move. My last patches should be in my proof of
> > >> concept network namespace tree, if they don't show up elsewhere.
> > >
> > > Is that the tree I'd get from
> > >
> > > git-fetch
> > > git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
> > > master:ebieder.master
> > >
> > Yes.
> >
> > > ? So I'd add a user_ns to the struct sysfs_tag_info?
> > >
> > > If so I'll give it a whirl.
> > >
> > Sounds good. My apologies I keep being almost on the verge
> > of getting someplace.
>
> Ok I've got the sysfs relevant patches ported to 2.6.25, and am looking
> at how to extend it to handle /sys/kernel/uids. You have tagging tied
> intimately to struct class. So the question is should I generalize the
> taggint to deal with kobjects instead, or create a struct class user
> and make /sys/kernel/uids a symlink to /sys/class/user/uids?

Heh, never mind, I was thinking class was a kobject class, not a device
class :) So I'll just have to generalize tagging.

thanks,
-serge

> Opinions?

>
> thanks,
> -serge
>
> PS: If you want me to post the patchset before handling userns I can do
> that, otherwise I was just going to wait until I'm done with userns.

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by ebiederm on Wed, 23 Apr 2008 18:49:30 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting Serge E. Hallyn (serue@us.ibm.com):
>> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
>> > >>
>> > >> I'm hoping to be able to get back at this in the week or so as things
>> > >> settle down from my move. My last patches should be in my proof of
>> > >> concept network namespace tree, if they don't show up elsewhere.
>> > >
>> > > Is that the tree I'd get from
>> > >
>> > > git-fetch
>> > > git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
>> > > master:ebieder.master
>> >
>> > Yes.
>> >
>> > > ? So I'd add a user_ns to the struct sysfs_tag_info?
>> > >
>> > > If so I'll give it a whirl.
>> >
>> > Sounds good. My apologies I keep being almost on the verge
>> > of getting someplace.
>>
>> Ok I've got the sysfs relevant patches ported to 2.6.25, and am looking
>> at how to extend it to handle /sys/kernel/uids. You have tagging tied
>> intimately to struct class. So the question is should I generalize the
>> taggint to deal with kobjects instead, or create a struct class user
>> and make /sys/kernel/uids a symlink to /sys/class/user/uids?
>
> Heh, never mind, I was thinking class was a kobject class, not a device

> class :) So I'll just have to generalize tagging.

Yes. You just need a way to get the tags there.

At the level of sysfs it is fairly general.
Getting through the kobject layer is a different story.

I suspect since you are working on this and I seem to be stuck
in molasses at the moment it makes sense to figure out what it
will take to handle the uid namespace before pushing these
patches again.

Taking a quick look and having a clue what we will need to
do for a theoretical device namespace is also a possibility.

Eric

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Fri, 25 Apr 2008 19:21:02 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > Quoting Serge E. Hallyn (serue@us.ibm.com):
> >> Quoting Eric W. Biederman (ebiederm@xmission.com):
> >> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >> >>
> >> >> I'm hoping to be able to get back at this in the week or so as things
> >> >> settle down from my move. My last patches should be in my proof of
> >> >> concept network namespace tree, if they don't show up elsewhere.
> >> >
> >> > Is that the tree I'd get from
> >> >
> >> > git-fetch
> >> > git://git.kernel.org/pub/scm/linux/kernel/git/ebiederm/linux-2.6-netns.git
> >> > master:ebieder.master
> >>
> >> Yes.
> >>
> >> > ? So I'd add a user_ns to the struct sysfs_tag_info?
> >> >
> >> > If so I'll give it a whirl.

> >> >
> >> > Sounds good. My apologies I keep being almost on the verge
> >> > of getting someplace.
> >>
> >> Ok I've got the sysfs relevant patches ported to 2.6.25, and am looking
> >> at how to extend it to handle /sys/kernel/uids. You have tagging tied
> >> intimately to struct class. So the question is should I generalize the
> >> taggint to deal with kobjects instead, or create a struct class user
> >> and make /sys/kernel/uids a symlink to /sys/class/user/uids?
> >
> > Heh, never mind, I was thinking class was a kobject class, not a device
> > class :) So I'll just have to generalize tagging.
>
> Yes. You just need a way to get the tags there.
>
> At the level of sysfs it is fairly general.
> Getting through the kobject layer is a different story.

Heh, well I tried several approaches - adding tag_ops to kset, to ktype,
etc. Finally ended up just calling sysfs_enable_tagging on
/sys/kernel/uids when that is created. It's now working perfectly.

> I suspect since you are working on this and I seem to be stuck
> in molasses at the moment it makes sense to figure out what it
> will take to handle the uid namespace before pushing these
> patches again.

I had ported your patches to 2.6.25, but Benjamin in the meantime ported
them to 2.6.25-mm1. Since that's closer to the -net tree it's a more
useful port, so I'll let him post his patchset. Then I'll send the
userns patch on top of that. While I'm not actually able to send
network traffic over a veth dev (I probably am still not setting it up
right), I am able to pass veth devices into network namespaces, and the
user namespaces are properly handled.

I believe Benjamin did notice a problem with some symlinks not existing,
and I think we want one more patch on top of yours removing the
hold_net() from sysfs_mount, which I don't think was what you really
wanted to do. By simply removing that, if all tasks in a netns go away,
the netns actually goes away and a lookup under a bind-mounted copy of
its /sys/class/net is empty.

Anyway the patches should be hitting the list next week.

> Taking a quick look and having a clue what we will need to
> do for a theoretical device namespace is also a possibility.

I'm not sure I'm familiar enough with the kobject/class/sysfs/device

relationships yet to comment on that. It doesn't look like it should
really be a problem, though simply adding tags to every directory
under /sys/class (/sys/class/tty, /sys/class/usb_device, etc) doesn't
seem like necessarily the nicest way to go...

thanks,
-serge

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by ebiederm on Fri, 25 Apr 2008 19:47:27 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Heh, well I tried several approaches - adding tag_ops to kset, to ktype,
> etc. Finally ended up just calling sysfs_enable_tagging on
> /sys/kernel/uids when that is created. It's now working perfectly.

Sounds good.

>> I suspect since you are working on this and I seem to be stuck
>> in molasses at the moment it makes sense to figure out what it
>> will take to handle the uid namespace before pushing these
>> patches again.
>
> I had ported your patches to 2.6.25, but Benjamin in the meantime ported
> them to 2.6.25-mm1. Since that's closer to the -net tree it's a more
> useful port, so I'll let him post his patchset. Then I'll send the
> userns patch on top of that. While I'm not actually able to send
> network traffic over a veth dev (I probably am still not setting it up
> right), I am able to pass veth devices into network namespaces, and the
> user namespaces are properly handled.
>
> I believe Benjamin did notice a problem with some symlinks not existing,
> and I think we want one more patch on top of yours removing the
> hold_net() from sysfs_mount, which I don't think was what you really
> wanted to do. By simply removing that, if all tasks in a netns go away,
> the netns actually goes away and a lookup under a bind-mounted copy of
> its /sys/class/net is empty.

I will have to look, I need to refresh myself on where all of this code is.
I think hold_net was what I wanted. A record that there is a user

but not something that will keep the network namespace from going away.

Essentially hold_net should be a debugging check rather then a
real limitation.

> Anyway the patches should be hitting the list next week.

Cool. We can figure out what we need to do to merge them from
there.

>> Taking a quick look and having a clue what we will need to
>> do for a theoretical device namespace is also a possibility.
>
> I'm not sure I'm familiar enough with the kobject/class/sysfs/device
> relationships yet to comment on that. It doesn't look like it should
> really be a problem, though simply adding tags to every directory
> under /sys/class (/sys/class/tty, /sys/class/usb_device, etc) doesn't
> seem like necessarily the nicest way to go...

True. And the goal is something maintainable. There are still a lot
of implications of a device namespace left unexamined so we shall see.

Eric

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

---

Subject: Re: [RFC][PATCH 0/7] Clone PTS namespace
Posted by serue on Sat, 26 Apr 2008 13:02:43 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
>
> > Heh, well I tried several approaches - adding tag_ops to kset, to ktype,
> > etc. Finally ended up just calling sysfs_enable_tagging on
> > /sys/kernel/uids when that is created. It's now working perfectly.
>
> Sounds good.
>
> >> I suspect since you are working on this and I seem to be stuck
> >> in molasses at the moment it makes sense to figure out what it
> >> will take to handle the uid namespace before pushing these
> >> patches again.

> >
> > I had ported your patches to 2.6.25, but Benjamin in the meantime ported
> > them to 2.6.25-mm1. Since that's closer to the -net tree it's a more
> > useful port, so I'll let him post his patchset. Then I'll send the
> > userns patch on top of that. While I'm not actually able to send
> > network traffic over a veth dev (I probably am still not setting it up
> > right), I am able to pass veth devices into network namespaces, and the
> > user namespaces are properly handled.
> >
> > I believe Benjamin did notice a problem with some symlinks not existing,
> > and I think we want one more patch on top of yours removing the
> > hold_net() from sysfs_mount, which I don't think was what you really
> > wanted to do. By simply removing that, if all tasks in a netns go away,
> > the netns actually goes away and a lookup under a bind-mounted copy of
> > its /sys/class/net is empty.
>
> I will have to look, I need to refresh myself on where all of this code is.
> I think hold_net was what I wanted. A record that there is a user
> but not something that will keep the network namespace from going away.
>
> Essentially hold_net should be a debugging check rather then a
> real limitation.

Ah, I see, I assumed it actually pinned it. Sorry, never mind then :)

-serge

> > Anyway the patches should be hitting the list next week.
>
> Cool. We can figure out what we need to do to merge them from
> there.
>
> >> Taking a quick look and having a clue what we will need to
> >> do for a theoretical device namespace is also a possibility.
> >
> > I'm not sure I'm familiar enough with the kobject/class/sysfs/device
> > relationships yet to comment on that. It doesn't look like it should
> > really be a problem, though simply adding tags to every directory
> > under /sys/class (/sys/class/tty, /sys/class/usb_device, etc) doesn't
> > seem like necessarily the nicest way to go...
>
> True. And the goal is something maintainable. There are still a lot
> of implications of a device namespace left unexamined so we shall see.
>
> Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org