Subject: Re: [RFC][PATCH 14/16] Introduce proc_mnt for pid_ns
Posted by xemul on Thu, 24 May 2007 09:23:01 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Subject: Introduce proc_mnt for pid_ns
>
> From: Dave Hansen <hansendc@us.ibm.com>
>
> The following patch completes the removal of the global proc_mnt.
> It fetches the mnt on which to do dentry invalidations from the
> pid_namespace in which the task appears.
>
> For now, there is only one pid namespace in mainline so this is
> straightforward. In the -lxc tree we'll have to do something
> more complex. The proc_flush_task() code takes a task, and
> needs to be able to find the corresponding proc superblocks on
> which that tasks's /proc/<pid> directories could appear. We
> can tell in which pid namespaces a task appears, so I put a
> pointer from the pid namespace to the corresponding proc_mnt.
>
> /proc currently has some special code to make sure that the root
> directory gets set up correctly. It proc_mnt variable in order
> to find its way to the root inode.
>
> Signed-off-by: Dave Hansen <haveblue@us.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
>
> fs/proc/base.c | 32 +++++++++++++++++++++++++-
> fs/proc/inode.c | 11 +++++++-
> fs/proc/root.c | 52 ++++++++++++++++++++++++++++++--------------
> include/linux/pid_namespace.h | 1
> include/linux/proc_fs.h | 1
> 5 files changed, 75 insertions(+), 22 deletions(-)
>
> Index: lx26-21-mm2/fs/proc/base.c
> ===
> --- lx26-21-mm2.orig/fs/proc/base.c 2007-05-22 16:59:49.000000000 -0700
> +++ lx26-21-mm2/fs/proc/base.c 2007-05-22 16:59:56.000000000 -0700
> @@ -2005,9 +2005,11 @@ static const struct inode_operations pro
> };
>
> /**
> - * proc_flush_task - Remove dcache entries for @task from the /proc dcache.
> + * proc_flush_task_from_pid_ns - Remove dcache entries for @task
> + *   from the /proc dcache.
>   *

```
>   * @task: task that should be flushed.
> + * @pid_ns: pid_namespace in which that task appears
>   *
>   * Looks in the dcache for
>   * /proc/@pid
> @@ -2025,11 +2027,22 @@ static const struct inode_operations pro
>   *       that no dcache entries will exist at process exit time it
>   *       just makes it very unlikely that any will persist.
>   */
> -void proc_flush_task(struct task_struct *task)
> +static void proc_flush_task_from_pid_ns(struct task_struct *task,
> +    struct pid_namespace* pid_ns)
>  {
>   struct dentry *dentry, *leader, *dir;
>   char buf[PROC_NUMBUF];
>   struct qstr name;
> + struct vfsmount *proc_mnt;
> +
> + WARN_ON(!pid_ns);
> + /*
> +  * It is possible that no /procs have been instantiated
> +  * for this particular pid namespace.
> +  */
> + if (!pid_ns->proc_mnt)
> +  return;
> + proc_mnt = pid_ns->proc_mnt;
>
>   name.name = buf;
>   name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
> @@ -2071,6 +2084,21 @@ out:
>   return;
>  }
>
> +void proc_flush_task(struct task_struct *task)
> +{
> + int i;
> + struct pid *pid;
> + struct upid* upid;
> +
> + pid = task_pid(task);
> + if (!pid)
> +  return;
> +
> + upid = &pid->upid_list[0];
> + for (i = 0; i < pid->num_upids; i++, upid++)
> +  proc_flush_task_from_pid_ns(task, upid->pid_ns);
> +}
> +
```

> static struct dentry *proc_pid_instantiate(struct inode *dir,
> struct dentry * dentry,
> struct task_struct *task, const void *ptr)
> Index: lx26-21-mm2/fs/proc/inode.c
> ===
> --- lx26-21-mm2.orig/fs/proc/inode.c 2007-05-22 16:51:19.000000000 -0700
> +++ lx26-21-mm2/fs/proc/inode.c 2007-05-22 16:59:56.000000000 -0700
> @@ -6,6 +6,7 @@
>
> #include <linux/time.h>
> #include <linux/proc_fs.h>
> +#include <linux/hardirq.h>
> #include <linux/kernel.h>
> #include <linux/mm.h>
> #include <linux/string.h>
> @@ -74,8 +75,6 @@ static void proc_delete_inode(struct ino
> clear_inode(inode);
> }
>
> -struct vfsmount *proc_mnt;
> -
> static void proc_read_inode(struct inode * inode)
> {
> inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
> @@ -458,6 +457,8 @@ out_mod:
>
> int proc_fill_super(struct super_block *s, void *data, int silent)
> {
> + struct pid_namespace *pid_ns = data;
> + struct proc_inode *ei;
> struct inode * root_inode;
>
> s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
> @@ -466,6 +467,7 @@ int proc_fill_super(struct super_block *
> s->s_magic = PROC_SUPER_MAGIC;
> s->s_op = &proc_sops;
> s->s_time_gran = 1;
> + s->s_fs_info = pid_ns;

This must be done in set callback for sget. Otherwise we
may have two superblocks with the same pid_ns...

> de_get(&proc_root);
> root_inode = proc_get_inode(s, PROC_ROOT_INO, &proc_root);
> @@ -476,6 +478,11 @@ int proc_fill_super(struct super_block *
> s->s_root = d_alloc_root(root_inode);
> if (!s->s_root)
> goto out_no_root;

```
> + /* Seed the root directory with a pid so it doesn't need
> +  * to be special in base.c.
> +  */
> + ei = PROC_I(root_inode);
> + ei->pid = find_get_pid(1);
>   return 0;
>
>  out_no_root:
> Index: lx26-21-mm2/fs/proc/root.c
> ===================================================================
> --- lx26-21-mm2.orig/fs/proc/root.c 2007-05-22 16:51:19.000000000 -0700
> +++ lx26-21-mm2/fs/proc/root.c 2007-05-22 16:59:56.000000000 -0700
> @@ -12,32 +12,56 @@
>  #include <linux/time.h>
>  #include <linux/proc_fs.h>
>  #include <linux/stat.h>
> +#include <linux/hardirq.h>
>  #include <linux/init.h>
>  #include <linux/sched.h>
>  #include <linux/module.h>
>  #include <linux/bitops.h>
>  #include <linux/smp_lock.h>
>  #include <linux/mount.h>
> +#include <linux/pid_namespace.h>
>
>  #include "internal.h"
>
>  struct proc_dir_entry *proc_net, *proc_net_stat, *proc_bus, *proc_root_fs, *proc_root_driver;
>
> +static int proc_test_sb(struct super_block *s, void *data)
> +{
> + struct pid_namespace *pid_ns = data;
> + if (s->s_fs_info == pid_ns)
> +  return 1;
> + return 0;
> +}
> +
>  static int proc_get_sb(struct file_system_type *fs_type,
>   int flags, const char *dev_name, void *data, struct vfsmount *mnt)
>  {
> - if (proc_mnt) {
> -  /* Seed the root directory with a pid so it doesn't need
> -   * to be special in base.c.  I would do this earlier but
> -   * the only task alive when /proc is mounted the first time
> -   * is the init_task and it doesn't have any pids.
> -   */
> -  struct proc_inode *ei;
> -  ei = PROC_I(proc_mnt->mnt_sb->s_root->d_inode);
```

```
> -  if (!ei->pid)
> -   ei->pid = find_get_pid(1);
> + int error;
> + struct super_block *s;
> + struct pid_namespace *pid_ns;
> +
> + /*
> +  * We can eventually derive this out of whatever mount
> +  * arguments the user supplies, but just take it from
> +  * current for now.
> +  */
> + pid_ns = task_active_pid_ns(current);
> +
> + s = sget(fs_type, proc_test_sb, set_anon_super, pid_ns);
> + if (IS_ERR(s))
> +  return PTR_ERR(s);
> +
> + error = proc_fill_super(s, pid_ns, 0);
> + if (error) {
> +  deactivate_super(s);
> +  return error;
>   }
> - return get_sb_single(fs_type, flags, data, proc_fill_super, mnt);
> +
> + if (!pid_ns->proc_mnt)
> +  pid_ns->proc_mnt = mnt;
> +
> + do_remount_sb(s, flags, data, 0);
> + return simple_set_mnt(mnt, s);
>  }
>
>  static struct file_system_type proc_fs_type = {
> @@ -54,12 +78,6 @@ void __init proc_root_init(void)
>   err = register_filesystem(&proc_fs_type);
>   if (err)
>    return;
> - proc_mnt = kern_mount(&proc_fs_type);
> - err = PTR_ERR(proc_mnt);
> - if (IS_ERR(proc_mnt)) {
> -  unregister_filesystem(&proc_fs_type);
> -  return;
> - }
>   proc_misc_init();
>   proc_net = proc_mkdir("net", NULL);
>   proc_net_stat = proc_mkdir("net/stat", NULL);
> Index: lx26-21-mm2/include/linux/pid_namespace.h
> ===================================================================
> --- lx26-21-mm2.orig/include/linux/pid_namespace.h 2007-05-22 16:59:53.000000000 -0700
```

> +++ lx26-21-mm2/include/linux/pid_namespace.h 2007-05-22 16:59:56.000000000 -0700
> @@ -33,6 +33,7 @@ struct pid_namespace {
> int last_pid;
> struct task_struct *child_reaper;
> atomic_t terminating;
> + struct vfsmount *proc_mnt;
> };
>
> extern struct pid_namespace init_pid_ns;
> Index: lx26-21-mm2/include/linux/proc_fs.h
> ===
> --- lx26-21-mm2.orig/include/linux/proc_fs.h 2007-05-22 16:51:19.000000000 -0700
> +++ lx26-21-mm2/include/linux/proc_fs.h 2007-05-22 16:59:56.000000000 -0700
> @@ -125,7 +125,6 @@ extern struct proc_dir_entry *create_pro
> struct proc_dir_entry *parent);
> extern void remove_proc_entry(const char *name, struct proc_dir_entry *parent);
>
> -extern struct vfsmount *proc_mnt;
> extern int proc_fill_super(struct super_block *,void *,int);
> extern struct inode *proc_get_inode(struct super_block *, unsigned int, struct proc_dir_entry *);
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

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