Subject: Re: [RFC][PATCH] Pid namespaces vs locks interaction
Posted by serue on Thu, 06 Dec 2007 14:53:40 GMT
View Forum Message <> Reply to Message

Quoting Vitaliy Gusev (vgusev@openvz.org):
> Hello!
>
> I am working on pid namespaces vs locks interaction and want to evaluate the
> idea.
> fcntl(F_GETLK,..) can return pid of process for not current pid namespace (if
> process is belonged to the several namespaces). It is true also for pids
> in /proc/locks. So correct behavior is saving pointer to the struct pid of
> the process lock owner.
> --
> Thank,
> Vitaliy Gusev

> diff --git a/fs/locks.c b/fs/locks.c
> index 8b8388e..d2d3d75 100644
> --- a/fs/locks.c
> +++ b/fs/locks.c
> @@ -125,6 +125,7 @@
> #include <linux/syscalls.h>
> #include <linux/time.h>
> #include <linux/rcupdate.h>
> +#include <linux/pid_namespace.h>
>
> #include <asm/semaphore.h>
> #include <asm/uaccess.h>
> @@ -185,6 +186,7 @@ void locks_init_lock(struct file_lock *fl)
>   fl->fl_fasync = NULL;
>   fl->fl_owner = NULL;
>   fl->fl_pid = 0;
> + fl->fl_nspid = NULL;

The idea seems right, but why are you keeping fl->fl_pid around?

Seems like the safer thing to do would be to have a separate
struct user_flock, with an integer pid, for communicating to userspace,
and a struct flock, with struct pid, for kernel use? Then fcntl_getlk()
and fcntl_setlk() do the appropriate conversions.

thanks,
-serge

>   fl->fl_file = NULL;
>   fl->fl_flags = 0;
>   fl->fl_type = 0;

```
> @@ -553,6 +555,8 @@ static void locks_insert_lock(struct file_lock **pos, struct file_lock *fl)
>  {
>   list_add(&fl->fl_link, &file_lock_list);
>
> + fl->fl_nspid = get_pid(task_tgid(current));
> +
>   /* insert into file's list */
>   fl->fl_next = *pos;
>   *pos = fl;
> @@ -584,6 +588,11 @@ static void locks_delete_lock(struct file_lock **thisfl_p)
>   if (fl->fl_ops && fl->fl_ops->fl_remove)
>    fl->fl_ops->fl_remove(fl);
>
> + if (fl->fl_nspid) {
> +  put_pid(fl->fl_nspid);
> +  fl->fl_nspid = NULL;
> + }
> +
>   locks_wake_up_blocks(fl);
>   locks_free_lock(fl);
>  }
> @@ -673,14 +682,16 @@ posix_test_lock(struct file *filp, struct file_lock *fl)
>    if (posix_locks_conflict(fl, cfl))
>     break;
>   }
> - if (cfl)
> + if (cfl) {
>    __locks_copy_lock(fl, cfl);
> - else
> +  if (cfl->fl_nspid)
> +   fl->fl_pid = pid_nr_ns(cfl->fl_nspid,
> +       task_active_pid_ns(current));
> + } else
>    fl->fl_type = F_UNLCK;
>   unlock_kernel();
>   return;
>  }
> -
>  EXPORT_SYMBOL(posix_test_lock);
>
>  /* This function tests for deadlock condition before putting a process to
> @@ -2084,6 +2095,12 @@ static void lock_get_status(struct seq_file *f, struct file_lock *fl,
>         int id, char *pfx)
>  {
>   struct inode *inode = NULL;
> + unsigned int fl_pid;
> +
> + if (fl->fl_nspid)
```

```
> +  fl_pid = pid_nr_ns(fl->fl_nspid, task_active_pid_ns(current));
> + else
> +  fl_pid = fl->fl_pid;
>
>   if (fl->fl_file != NULL)
>    inode = fl->fl_file->f_path.dentry->d_inode;
> @@ -2124,16 +2141,16 @@ static void lock_get_status(struct seq_file *f, struct file_lock *fl,
>   }
>   if (inode) {
>  #ifdef WE_CAN_BREAK_LSLK_NOW
> -  seq_printf(f, "%d %s:%ld ", fl->fl_pid,
> +  seq_printf(f, "%d %s:%ld ", fl_pid,
>      inode->i_sb->s_id, inode->i_ino);
>  #else
>    /* userspace relies on this representation of dev_t ;-( */
> -  seq_printf(f, "%d %02x:%02x:%ld ", fl->fl_pid,
> +  seq_printf(f, "%d %02x:%02x:%ld ", fl_pid,
>      MAJOR(inode->i_sb->s_dev),
>      MINOR(inode->i_sb->s_dev), inode->i_ino);
>  #endif
>   } else {
> -  seq_printf(f, "%d <none>:0 ", fl->fl_pid);
> +  seq_printf(f, "%d <none>:0 ", fl_pid);
>   }
>   if (IS_POSIX(fl)) {
>    if (fl->fl_end == OFFSET_MAX)
> diff --git a/include/linux/fs.h b/include/linux/fs.h
> index b3ec4a4..5876f68 100644
> --- a/include/linux/fs.h
> +++ b/include/linux/fs.h
> @@ -870,6 +870,7 @@ struct file_lock {
>   struct list_head fl_block; /* circular list of blocked processes */
>   fl_owner_t fl_owner;
>   unsigned int fl_pid;
> + struct pid *fl_nspid;
>   wait_queue_head_t fl_wait;
>   struct file *fl_file;
>   unsigned char fl_flags;
```