
Subject: [PATCH 1/1][3rd resend] sys_unshare: remove the dead
CLONE_THREAD/SIGHAND/VM code

Posted by [Oleg Nesterov](#) on Tue, 15 Feb 2011 19:17:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

Cleanup: kill the dead code which does nothing but complicates the code
and confuses the reader.

sys_unshare(CLONE_THREAD/SIGHAND/VM) is not really implemented, and I doubt
very much it will ever work. At least, nobody even tried since the original
"unshare system call -v5: system call handler function" commit
99d1419d96d7df9cfa56bc977810be831bd5ef64 was applied more than 4 years ago.

And the code is not consistent. unshare_thread() always fails unconditionally,
while unshare_sighand() and unshare_vm() pretend to work if there is nothing
to unshare.

Remove unshare_thread(), unshare_sighand(), unshare_vm() helpers and related
variables and add a simple CLONE_THREAD | CLONE_SIGHAND | CLONE_VM check into
check_unshare_flags().

Also, move the "CLONE_NEWNS needs CLONE_FS" check from check_unshare_flags()
to sys_unshare(). This looks more consistent and matches the similar
do_sysvsem check in sys_unshare().

Note: with or without this patch "atomic_read(mm->mm_users) > 1" can give
a false positive due to get_task_mm().

Signed-off-by: Oleg Nesterov <oleg@redhat.com>

Acked-by: Roland McGrath <roland@redhat.com>

kernel/fork.c | 123 ++++++-----
1 file changed, 25 insertions(+), 98 deletions(-)

--- 2.6.37/kernel/fork.c~unshare-killcrap 2010-11-05 18:03:28.000000000 +0100

+++ 2.6.37/kernel/fork.c 2010-11-05 18:09:52.000000000 +0100

@@ -1522,38 +1522,24 @@ void __init proc_caches_init(void)

}

/*

- * Check constraints on flags passed to the unshare system call and

- * force unsharing of additional process context as appropriate.

+ * Check constraints on flags passed to the unshare system call.

*/

-static void check_unshare_flags(unsigned long *flags_ptr)

+static int check_unshare_flags(unsigned long unshare_flags)

{

```

+ if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
+   CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
+   CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWNET))
+ return -EINVAL;
/*
- * If unsharing a thread from a thread group, must also
- * unshare vm.
- */
- if (*flags_ptr & CLONE_THREAD)
- *flags_ptr |= CLONE_VM;
-
- /*
- * If unsharing vm, must also unshare signal handlers.
- */
- if (*flags_ptr & CLONE_VM)
- *flags_ptr |= CLONE_SIGHAND;
-
- /*
- * If unsharing namespace, must also unshare filesystem information.
+ * Not implemented, but pretend it works if there is nothing to
+ * unshare. Note that unsharing CLONE_THREAD or CLONE_SIGHAND
+ * needs to unshare vm.
+ */
- if (*flags_ptr & CLONE_NEWNS)
- *flags_ptr |= CLONE_FS;
-}
-
-/*
- * Unsharing of tasks created with CLONE_THREAD is not supported yet
- */
-static int unshare_thread(unsigned long unshare_flags)
-{
- if (unshare_flags & CLONE_THREAD)
- return -EINVAL;
+ if (unshare_flags & (CLONE_THREAD | CLONE_SIGHAND | CLONE_VM)) {
+ /* FIXME: get_task_mm() increments ->mm_users */
+ if (atomic_read(&current->mm->mm_users) > 1)
+ return -EINVAL;
+ }

    return 0;
}
@@ -1580,34 +1566,6 @@ static int unshare_fs(unsigned long unsh
}

/*
- * Unsharing of sighand is not supported yet
- */

```

```

-static int unshare_sighand(unsigned long unshare_flags, struct sighand_struct **new_sighp)
-{
- struct sighand_struct *sigh = current->sighand;
-
- if ((unshare_flags & CLONE_SIGHAND) && atomic_read(&sigh->count) > 1)
- return -EINVAL;
- else
- return 0;
-}
-
-/*
- * Unshare vm if it is being shared
- */
-static int unshare_vm(unsigned long unshare_flags, struct mm_struct **new_mmp)
-{
- struct mm_struct *mm = current->mm;
-
- if ((unshare_flags & CLONE_VM) &&
-     (mm && atomic_read(&mm->mm_users) > 1)) {
- return -EINVAL;
- }
-
- return 0;
-}
-
-/*
- * Unshare file descriptor table if it is being shared
- */
static int unshare_fd(unsigned long unshare_flags, struct files_struct **new_fdp)
@@ -1635,45 +1593,37 @@ static int unshare_fd(unsigned long unsh
-/*
- SYSCALL_DEFINE1(unshare, unsigned long, unshare_flags)
- {
- int err = 0;
- struct fs_struct *fs, *new_fs = NULL;
- struct sighand_struct *new_sigh = NULL;
- struct mm_struct *mm, *new_mm = NULL, *active_mm = NULL;
- struct files_struct *fd, *new_fd = NULL;
- struct nsproxy *new_nsproxy = NULL;
- int do_sysvsem = 0;
+ int err;

- check_unshare_flags(&unshare_flags);
-
- /* Return -EINVAL for all unsupported flags */
- err = -EINVAL;
- if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
- CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|

```

```

- CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWNET))
+ err = check_unshare_flags(unshare_flags);
+ if (err)
    goto bad_unshare_out;

/*
+ * If unsharing namespace, must also unshare filesystem information.
+ */
+ if (unshare_flags & CLONE_NEWNS)
+ unshare_flags |= CLONE_FS;
+ /*
    * CLONE_NEWIPC must also detach from the undolist: after switching
    * to a new ipc namespace, the semaphore arrays from the old
    * namespace are unreachable.
    */
    if (unshare_flags & (CLONE_NEWIPC|CLONE_SYSVSEM))
        do_sysvsem = 1;
- if ((err = unshare_thread(unshare_flags)))
- goto bad_unshare_out;
    if ((err = unshare_fs(unshare_flags, &new_fs)))
- goto bad_unshare_cleanup_thread;
- if ((err = unshare_sighand(unshare_flags, &new_sigh)))
- goto bad_unshare_cleanup_fs;
- if ((err = unshare_vm(unshare_flags, &new_mm)))
- goto bad_unshare_cleanup_sigh;
+ goto bad_unshare_out;
    if ((err = unshare_fd(unshare_flags, &new_fd)))
- goto bad_unshare_cleanup_vm;
+ goto bad_unshare_cleanup_fs;
    if ((err = unshare_nsproxy_namespaces(unshare_flags, &new_nsproxy,
        new_fs)))
        goto bad_unshare_cleanup_fd;

- if (new_fs || new_mm || new_fd || do_sysvsem || new_nsproxy) {
+ if (new_fs || new_fd || do_sysvsem || new_nsproxy) {
    if (do_sysvsem) {
        /*
        * CLONE_SYSVSEM is equivalent to sys_exit().
        @@ -1699,19 +1649,6 @@ SYSCALL_DEFINE1(unshare, unsigned long,
            spin_unlock(&fs->lock);
        }

- if (new_mm) {
- mm = current->mm;
- active_mm = current->active_mm;
- current->mm = new_mm;
- current->active_mm = new_mm;
- if (current->signal->oom_score_adj == OOM_SCORE_ADJ_MIN) {

```

```

- atomic_dec(&mm->oom_disable_count);
- atomic_inc(&new_mm->oom_disable_count);
- }
- activate_mm(active_mm, new_mm);
- new_mm = mm;
- }
-
  if (new_fd) {
    fd = current->files;
    current->files = new_fd;
@@ -1728,20 +1665,10 @@ bad_unshare_cleanup_fd:
  if (new_fd)
    put_files_struct(new_fd);

-bad_unshare_cleanup_vm:
- if (new_mm)
-  mmput(new_mm);
-
-bad_unshare_cleanup_sigh:
- if (new_sigh)
-  if (atomic_dec_and_test(&new_sigh->count))
-   kmem_cache_free(sighand_cache, new_sigh);
-
bad_unshare_cleanup_fs:
  if (new_fs)
    free_fs_struct(new_fs);

-bad_unshare_cleanup_thread:
bad_unshare_out:
  return err;
}

```

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