

---

Subject: - use-struct-pid-parameter-in-copy\_process.patch removed from -mm tree  
Posted by [akpm](#) on Sat, 12 May 2007 05:34:35 GMT

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

---

The patch titled

Use struct pid parameter in copy\_process()  
has been removed from the -mm tree. Its filename was  
use-struct-pid-parameter-in-copy\_process.patch

This patch was dropped because it was merged into mainline or a subsystem tree

-----  
Subject: Use struct pid parameter in copy\_process()

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Modify copy\_process() to take a struct pid \* parameter instead of a pid\_t.  
This simplifies the code a bit and also avoids having to call find\_pid() to  
convert the pid\_t to a struct pid.

Changelog:

- Fixed Badari Pulavarty's comments and passed in &init\_struct\_pid from fork\_idle().
- Fixed Eric Biederman's comments and simplified this patch and used a new patch to remove the likely(pid) check.

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

Cc: Cedric Le Goater <clg@fr.ibm.com>

Cc: Dave Hansen <haveblue@us.ibm.com>

Cc: Serge Hallyn <serue@us.ibm.com>

Cc: Eric Biederman <ebiederm@xmission.com>

Cc: <containers@lists.osdl.org>

Acked-by: Eric W. Biederman <ebiederm@xmission.com>

Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

---

kernel/fork.c | 11 ++++++-----

1 file changed, 6 insertions(+), 5 deletions(-)

diff -puN kernel/fork.c~use-struct-pid-parameter-in-copy\_process kernel/fork.c

--- a/kernel/fork.c~use-struct-pid-parameter-in-copy\_process

+++ a/kernel/fork.c

```
@@ -956,7 +956,7 @@ static struct task_struct *copy_process(  
    unsigned long stack_size,  
    int __user *parent_tidptr,  
    int __user *child_tidptr,  
-    int pid)  
+    struct pid *pid)  
{
```

```

int retval;
struct task_struct *p = NULL;
@@ -1023,7 +1023,7 @@ static struct task_struct *copy_process(
    p->did_exec = 0;
    delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
    copy_flags(clone_flags, p);
- p->pid = pid;
+ p->pid = pid_nr(pid);
    retval = -EFAULT;
    if (clone_flags & CLONE_PARENT_SETTID)
        if (put_user(p->pid, parent_tidptr))
@@ -1261,7 +1261,7 @@ static struct task_struct *copy_process(
    list_add_tail_rcu(&p->tasks, &init_task.tasks);
    __get_cpu_var(process_counts)++;
}
- attach_pid(p, PIDTYPE_PID, find_pid(p->pid));
+ attach_pid(p, PIDTYPE_PID, pid);
    nr_threads++;
}

@@ -1325,7 +1325,8 @@ struct task_struct * __cpuinit fork_idle
    struct task_struct *task;
    struct pt_regs regs;

- task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL, 0);
+ task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL,
+ &init_struct_pid);
    if (!IS_ERR(task))
        init_idle(task, cpu);

@@ -1375,7 +1376,7 @@ long do_fork(unsigned long clone_flags,
    clone_flags |= CLONE_PTRACE;
}

- p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, nr);
+ p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
/*
 * Do this prior waking up the new thread - the thread pointer
 * might get invalid after that point, if the thread exits quickly.

```

Patches currently in -mm which might be from sukadev@us.ibm.com are

origin.patch

---

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

