
Subject: + attach_pid-with-struct-pid-parameter.patch added to -mm tree

Posted by [akpm](#) on Thu, 15 Mar 2007 19:54:29 GMT

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

The patch titled

attach_pid() with struct pid parameter
has been added to the -mm tree. Its filename is
attach_pid-with-struct-pid-parameter.patch

*** Remember to use Documentation/SubmitChecklist when testing your code ***

See <http://www.zip.com.au/~akpm/linux/patches/stuff/added-to-mm.txt> to find
out what to do about this

Subject: attach_pid() with struct pid parameter
From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

attach_pid() currently takes a pid_t and then uses find_pid() to find the
corresponding struct pid. Sometimes we already have the struct pid. We can
then skip find_pid() if attach_pid() were to take a struct pid parameter.

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

```
fs/exec.c      | 2 +-  
include/linux/pid.h | 3 +--  
kernel/exec.c  | 4 ++--  
kernel/fork.c  | 11 ++++++----  
kernel/pid.c   | 9 ++++++---  
kernel/sys.c   | 2 +-  
6 files changed, 18 insertions(+), 13 deletions(-)
```

diff -puN fs/exec.c~attach_pid-with-struct-pid-parameter fs/exec.c

--- a/fs/exec.c~attach_pid-with-struct-pid-parameter

+++ a/fs/exec.c

```
@@ -701,7 +701,7 @@ static int de_thread(struct task_struct  
    */  
    detach_pid(tsk, PIDTYPE_PID);  
    tsk->pid = leader->pid;  
- attach_pid(tsk, PIDTYPE_PID, tsk->pid);  
+ attach_pid(tsk, PIDTYPE_PID, find_pid(tsk->pid));  
    transfer_pid(leader, tsk, PIDTYPE_PGID);
```

```

    transfer_pid(leader, tsk, PIDTYPE_SID);
    list_replace_rcu(&leader->tasks, &tsk->tasks);
diff -puN include/linux/pid.h~attach_pid-with-struct-pid-parameter include/linux/pid.h
--- a/include/linux/pid.h~attach_pid-with-struct-pid-parameter
+++ a/include/linux/pid.h
@@ -76,8 +76,7 @@ extern struct pid *get_task_pid(struct t
 * write-held.
 */
extern int FASTCALL(attach_pid(struct task_struct *task,
- enum pid_type type, int nr));
-
+ enum pid_type type, struct pid *pid));
extern void FASTCALL(detach_pid(struct task_struct *task, enum pid_type));
extern void FASTCALL(transfer_pid(struct task_struct *old,
    struct task_struct *new, enum pid_type));
diff -puN kernel/exit.c~attach_pid-with-struct-pid-parameter kernel/exit.c
--- a/kernel/exit.c~attach_pid-with-struct-pid-parameter
+++ a/kernel/exit.c
@@ -299,12 +299,12 @@ void __set_special_pids(pid_t session, p
 if (process_session(curr) != session) {
    detach_pid(curr, PIDTYPE_SID);
    set_signal_session(curr->signal, session);
- attach_pid(curr, PIDTYPE_SID, session);
+ attach_pid(curr, PIDTYPE_SID, find_pid(session));
 }
 if (process_group(curr) != pgrp) {
    detach_pid(curr, PIDTYPE_PGID);
    curr->signal->pgrp = pgrp;
- attach_pid(curr, PIDTYPE_PGID, pgrp);
+ attach_pid(curr, PIDTYPE_PGID, find_pid(pgrp));
 }
 }

diff -puN kernel/fork.c~attach_pid-with-struct-pid-parameter kernel/fork.c
--- a/kernel/fork.c~attach_pid-with-struct-pid-parameter
+++ a/kernel/fork.c
@@ -1242,16 +1242,19 @@ static struct task_struct *copy_process(
    tracehook_init_task(p);

    if (thread_group_leader(p)) {
+ pid_t pgid = process_group(current);
+ pid_t sid = process_session(current);
+
    p->signal->tty = current->signal->tty;
- p->signal->pgrp = process_group(current);
+ p->signal->pgrp = pgid;
    set_signal_session(p->signal, process_session(current));
- attach_pid(p, PIDTYPE_PGID, process_group(p));

```

```
- attach_pid(p, PIDTYPE_SID, process_session(p));
+ attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
+ attach_pid(p, PIDTYPE_SID, find_pid(sid));
```

```
    list_add_tail_rcu(&p->tasks, &init_task.tasks);
    __get_cpu_var(process_counts)++;
}
- attach_pid(p, PIDTYPE_PID, p->pid);
+ attach_pid(p, PIDTYPE_PID, find_pid(p->pid));
  nr_threads++;
}
```

diff -puN kernel/pid.c~attach_pid-with-struct-pid-parameter kernel/pid.c

--- a/kernel/pid.c~attach_pid-with-struct-pid-parameter

+++ a/kernel/pid.c

```
@@ -247,13 +247,16 @@ struct pid *fastcall find_pid(int nr)
}
EXPORT_SYMBOL_GPL(find_pid);
```

```
-int fastcall attach_pid(struct task_struct *task, enum pid_type type, int nr)
+/*
```

```
+ * attach_pid() must be called with the tasklist_lock write-held.
```

```
+ */
```

```
+int fastcall attach_pid(struct task_struct *task, enum pid_type type,
+ struct pid *pid)
{
  struct pid_link *link;
- struct pid *pid;
```

```
  link = &task->pids[type];
- link->pid = pid = find_pid(nr);
+ link->pid = pid;
  hlist_add_head_rcu(&link->node, &pid->tasks[type]);
```

```
  return 0;
```

diff -puN kernel/sys.c~attach_pid-with-struct-pid-parameter kernel/sys.c

--- a/kernel/sys.c~attach_pid-with-struct-pid-parameter

+++ a/kernel/sys.c

```
@@ -1486,7 +1486,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
  if (process_group(p) != pgid) {
    detach_pid(p, PIDTYPE_PGID);
    p->signal->pgrp = pgid;
-   attach_pid(p, PIDTYPE_PGID, pgid);
+   attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
  }
```

```
  err = 0;
```

```
—
```

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

attach_pid-with-struct-pid-parameter.patch
statically-initialize-struct-pid-for-swapper.patch
explicitly-set-pgid-and-sid-of-init-process.patch
use-struct-pid-parameter-in-copy_process.patch
remove-the-likelypid-check-in-copy_process.patch
use-task_pgrp-task_session-in-copy_process.patch
kill-unused-session-and-group-values-in-rocket-driver.patch
fix-some-coding-style-errors-in-autofs.patch
replace-pid_t-in-autofs-with-struct-pid-reference.patch

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
