
Subject: [PATCH 10/12] pid: Replace do/while_each_task_pid with
do/while_each_pid_task

Posted by [ebiederm](#) on Wed, 13 Dec 2006 11:07:54 GMT

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

There isn't any real advantage to this change except that
it allows the old functions to be removed. Which is easier
on maintenance and puts the code in a more uniform style.

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

```
fs/ioprio.c      | 18 ++++++++-----
kernel/capability.c | 8 +++++--
kernel/sys.c     | 40 ++++++++-----
3 files changed, 41 insertions(+), 25 deletions(-)
```

```
diff --git a/fs/ioprio.c b/fs/ioprio.c
index 89e8da1..10d2c21 100644
```

```
--- a/fs/ioprio.c
```

```
+++ b/fs/ioprio.c
```

```
@@ -60,6 +60,7 @@ asmlinkage long sys_ioprio_set(int which, int who, int ioprio)
    int data = IOPRIO_PRIO_DATA(ioprio);
    struct task_struct *p, *g;
    struct user_struct *user;
+ struct pid *pgrp;
    int ret;
```

```
    switch (class) {
@@ -98,12 +99,14 @@ asmlinkage long sys_ioprio_set(int which, int who, int ioprio)
        break;
        case IOPRIO_WHO_PGRP:
            if (!who)
-       who = process_group(current);
-       do_each_task_pid(who, PIDTYPE_PGID, p) {
+       pgrp = task_pgrp(current);
+       else
+       pgrp = find_pid(who);
+       do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
                ret = set_task_ioprio(p, ioprio);
                if (ret)
                    break;
-       } while_each_task_pid(who, PIDTYPE_PGID, p);
+       } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
                break;
        case IOPRIO_WHO_USER:
            if (!who)
@@ -167,6 +170,7 @@ asmlinkage long sys_ioprio_get(int which, int who)
    {
```

```

    struct task_struct *g, *p;
    struct user_struct *user;
+ struct pid *pgrp;
    int ret = -ESRCH;
    int tmpio;

@@ -182,8 +186,10 @@ asmlinkage long sys_ioprio_get(int which, int who)
    break;
    case IOPRIO_WHO_PGRP:
        if (!who)
-        who = process_group(current);
-        do_each_task_pid(who, PIDTYPE_PGID, p) {
+        pgrp = task_pgrp(current);
+        else
+        pgrp = find_pid(who);
+        do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
            tmpio = get_task_ioprio(p);
            if (tmpio < 0)
                continue;
@@ -191,7 +197,7 @@ asmlinkage long sys_ioprio_get(int which, int who)
    ret = tmpio;
    else
        ret = ioprio_best(ret, tmpio);
-    } while_each_task_pid(who, PIDTYPE_PGID, p);
+    } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
    break;
    case IOPRIO_WHO_USER:
        if (!who)
diff --git a/kernel/capability.c b/kernel/capability.c
index edb845a..c8d3c77 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -92,15 +92,17 @@ out:
 * cap_set_pg - set capabilities for all processes in a given process
 * group. We call this holding task_capability_lock and tasklist_lock.
 */
-static inline int cap_set_pg(int pgrp, kernel_cap_t *effective,
+static inline int cap_set_pg(int pgrp_nr, kernel_cap_t *effective,
                             kernel_cap_t *inheritable,
                             kernel_cap_t *permitted)
{
    struct task_struct *g, *target;
    int ret = -EPERM;
    int found = 0;
+ struct pid *pgrp;

- do_each_task_pid(pgrp, PIDTYPE_PGID, g) {
+ pgrp = find_pid(pgrp_nr);

```

```

+ do_each_pid_task(pgrp, PIDTYPE_PGID, g) {
    target = g;
    while_each_thread(g, target) {
        if (!security_capset_check(target, effective,
@@ -113,7 +115,7 @@ static inline int cap_set_pg(int pgrp, kernel_cap_t *effective,
    }
    found = 1;
}
- } while_each_task_pid(pgrp, PIDTYPE_PGID, g);
+ } while_each_pid_task(pgrp, PIDTYPE_PGID, g);

    if (!found)
        ret = 0;
diff --git a/kernel/sys.c b/kernel/sys.c
index a8f2ebe..c452ba9 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -589,6 +589,7 @@ asmlinkage long sys_setpriority(int which, int who, int niceval)
    struct task_struct *g, *p;
    struct user_struct *user;
    int error = -EINVAL;
+ struct pid *pgrp;

    if (which > 2 || which < 0)
        goto out;
@@ -603,18 +604,21 @@ asmlinkage long sys_setpriority(int which, int who, int niceval)
    read_lock(&tasklist_lock);
    switch (which) {
        case PRIO_PROCESS:
-     if (!who)
-     who = current->pid;
-     p = find_task_by_pid(who);
+     if (who)
+     p = find_task_by_pid(who);
+     else
+     p = current;
        if (p)
            error = set_one_prio(p, niceval, error);
            break;
        case PRIO_PGRP:
-     if (!who)
-     who = process_group(current);
-     do_each_task_pid(who, PIDTYPE_PGID, p) {
+     if (who)
+     pgrp = find_pid(who);
+     else
+     pgrp = task_pgrp(current);
+     do_each_pid_task(pgrp, PIDTYPE_PGID, p) {

```

```

    error = set_one_prio(p, niceval, error);
- } while_each_task_pid(who, PIDTYPE_PGID, p);
+ } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
    break;
    case PRIO_USER:
        user = current->user;
@@ -649,6 +653,7 @@ asmlinkage long sys_getpriority(int which, int who)
    struct task_struct *g, *p;
    struct user_struct *user;
    long niceval, retval = -ESRCH;
+ struct pid *pgrp;

    if (which > 2 || which < 0)
        return -EINVAL;
@@ -656,9 +661,10 @@ asmlinkage long sys_getpriority(int which, int who)
    read_lock(&tasklist_lock);
    switch (which) {
        case PRIO_PROCESS:
- if (!who)
-     who = current->pid;
- p = find_task_by_pid(who);
+ if (who)
+     p = find_task_by_pid(who);
+ else
+     p = current;
        if (p) {
            niceval = 20 - task_nice(p);
            if (niceval > retval)
@@ -666,13 +672,15 @@ asmlinkage long sys_getpriority(int which, int who)
        }
        break;
        case PRIO_PGRP:
- if (!who)
-     who = process_group(current);
- do_each_task_pid(who, PIDTYPE_PGID, p) {
+ if (who)
+     pgrp = find_pid(who);
+ else
+     pgrp = task_pgrp(current);
+ do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
            niceval = 20 - task_nice(p);
            if (niceval > retval)
                retval = niceval;
- } while_each_task_pid(who, PIDTYPE_PGID, p);
+ } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
        break;
        case PRIO_USER:
            user = current->user;

```

@@ -1381,7 +1389,7 @@ asmlinkage long sys_setpgid(pid_t pid, pid_t pgid)

```
if (p->real_parent == group_leader) {  
    err = -EPERM;  
- if (process_session(p) != process_session(group_leader))  
+ if (task_session(p) != task_session(group_leader))  
    goto out;  
    err = -EACCES;  
    if (p->did_exec)
```

@@ -1400,7 +1408,7 @@ asmlinkage long sys_setpgid(pid_t pid, pid_t pgid)

```
    struct task_struct *g =  
        find_task_by_pid_type(PIDTYPE_PGID, pgid);  
  
- if (!g || process_session(g) != process_session(group_leader))  
+ if (!g || task_session(g) != task_session(group_leader))  
    goto out;  
}
```

--

1.4.4.1.g278f

Containers mailing list

Containers@lists.osdl.org

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