
Subject: [RFC][PATCH 4/4] PID: use the target ID specified in procfs
Posted by [Nadia Derby](#) on Mon, 10 Mar 2008 13:50:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

[PATCH 04/04]

This patch makes use of the target ids specified by a previous write to /proc/self/next_pids as the ids to use to allocate the next upid nrs. Upper levels upid nrs that are not specified in next_pids file are left to the kernel choice.

Signed-off-by: Nadia Derby <Nadia.Derbey@bull.net>

kernel/fork.c | 5 ++-
kernel/pid.c | 80 +++-----
2 files changed, 73 insertions(+), 12 deletions(-)

Index: linux-2.6.25-rc3-mm1/kernel/pid.c

```
=====
--- linux-2.6.25-rc3-mm1.orig/kernel/pid.c 2008-03-10 09:19:02.000000000 +0100
+++ linux-2.6.25-rc3-mm1/kernel/pid.c 2008-03-10 13:45:52.000000000 +0100
@@ -122,14 +122,26 @@ static void free_pidmap(struct upid *upi
     atomic_inc(&map->nr_free);
 }

-static int alloc_pidmap(struct pid_namespace *pid_ns)
+static int alloc_pidmap(struct pid_namespace *pid_ns, struct pid_list *pid_l,
+ int level)
 {
     int i, offset, max_scan, pid, last = pid_ns->last_pid;
     struct pidmap *map;

- pid = last + 1;
- if (pid >= pid_max)
- pid = RESERVED_PIDS;
+ if (!pid_l) {
+ pid = last + 1;
+ if (pid >= pid_max)
+ pid = RESERVED_PIDS;
+ } else {
+ /*
+  * There's a target pid, so use it instead
+  */
+ pid = PID_AT(pid_l, level);
+ if (pid >= pid_max)
+ return -EINVAL;

```

```

+ }
+
+ offset = pid & BITS_PER_PAGE_MASK;
+ map = &pid_ns->pidmap[pid/BITS_PER_PAGE];
+ max_scan = (pid_max + BITS_PER_PAGE - 1)/BITS_PER_PAGE - !offset;
@@ -153,9 +165,16 @@ static int alloc_pidmap(struct pid_names
+ do {
+     if (!test_and_set_bit(offset, map->page)) {
+         atomic_dec(&map->nr_free);
- pid_ns->last_pid = pid;
+ if (!pid_l)
+     pid_ns->last_pid = pid;
+ else
+     pid_ns->last_pid = max(last,
+         pid);
+     return pid;
+ }
+ if (pid_l)
+     /* Target pid is already in use */
+     return -EBUSY;
+     offset = find_next_offset(map, offset);
+     pid = mk_pid(pid_ns, map, offset);
+     /*
@@ -179,7 +198,7 @@ static int alloc_pidmap(struct pid_names
+ }
+     pid = mk_pid(pid_ns, map, offset);
+ }
- return -1;
+ return -ENOMEM;
+ }

int next_pidmap(struct pid_namespace *pid_ns, int last)
@@ -250,14 +269,55 @@ struct pid *alloc_pid(struct pid_namespa
+ int i, nr;
+ struct pid_namespace *tmp;
+ struct upid *upid;
+ struct pid_list *pid_l = NULL;
+ int rel_level = -1;
+
+ /*
+ * If there is a list of upid nrs specified, use it instead of letting
+ * the kernel chose them for us.
+ */
+ if (current->next_id && (current->next_id->flag & SYS_ID_PID)) {
+     pid_l = current->next_id->pid_ids;
+     BUG_ON(!pid_l);
+     rel_level = pid_l->npids - 1;
+     if (rel_level > ns->level) {

```

```

+ pid = ERR_PTR(-EINVAL);
+ goto out;
+ }
+ }

pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
- if (!pid)
+ if (!pid) {
+ pid = ERR_PTR(-ENOMEM);
goto out;
+ }

tmp = ns;
- for (i = ns->level; i >= 0; i--) {
- nr = alloc_pidmap(tmp);
+ /*
+ * Use the predefined upid nrs for levels ns->level down to
+ * ns->level - rel_level
+ */
+ for (i = ns->level; rel_level >= 0; i--, rel_level--) {
+ nr = alloc_pidmap(tmp, pid_l, rel_level);
+ if (nr < 0)
+ goto out_free;
+
+ pid->numbers[i].nr = nr;
+ pid->numbers[i].ns = tmp;
+ tmp = tmp->parent;
+ }
+
+ if (pid_l) {
+ current->next_id->flag &= ~SYS_ID_PID;
+ pids_free(pid_l);
+ current->next_id->pid_ids = NULL;
+ }
+
+ /*
+ * Let the lower levels upid nrs be automatically allocated
+ */
+ for (; i >= 0; i--) {
+ nr = alloc_pidmap(tmp, NULL, -1);
+ if (nr < 0)
+ goto out_free;

@@ -288,7 +348,7 @@ out_free:
free_pidmap(pid->numbers + i);

kmem_cache_free(ns->pid_cachep, pid);
- pid = NULL;

```

```
+ pid = ERR_PTR(nr);
  goto out;
}
```

Index: linux-2.6.25-rc3-mm1/kernel/fork.c

```
=====
--- linux-2.6.25-rc3-mm1.orig/kernel/fork.c 2008-03-10 09:19:01.000000000 +0100
+++ linux-2.6.25-rc3-mm1/kernel/fork.c 2008-03-10 13:48:03.000000000 +0100
@@ -1197,10 +1197,11 @@ static struct task_struct *copy_process(
     goto bad_fork_cleanup_io;

    if (pid != &init_struct_pid) {
-   retval = -ENOMEM;
    pid = alloc_pid(task_active_pid_ns(p));
-   if (!pid)
+   if (IS_ERR(pid)) {
+       retval = PTR_ERR(pid);
        goto bad_fork_cleanup_io;
+   }

    if (clone_flags & CLONE_NEWPID) {
        retval = pid_ns_prepare_proc(task_active_pid_ns(p));
    }
--
```

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