
Subject: [RFC][PATCH 1/3] Pid ns helpers for signals
Posted by [Sukadev Bhattiprolu](#) on Thu, 30 Aug 2007 06:19:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH] Pid namespace helpers for signals.

Define some helper functions that will be used to implement signal semantics with multiple pid namespaces.

ancestor_pid_ns(ns1, ns2):
TRUE if @ns1 is an ancestor of @ns2.

descendant_pid_ns(ns1, ns2):
TRUE if @ns1 is a descendant of @ns2.

(Note that "not an ancestor" does not mean descendant and vice versa - could be a cousin)

pid_ns_equal(tsk)
TRUE if active pid ns of @tsk is same as active pid ns of caller.

This patch also extends task_active_pid_ns() to try to return the pid ns of processes that have exited but not been reaped. Appreciate comments on that :-)

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

```
include/linux/pid.h      | 5 +++
include/linux/pid_namespace.h | 13 ++++++-
include/linux/sched.h    | 17 ++++++++
kernel/pid.c             | 64 ++++++++++++++++++++++++++++++++++++++
4 files changed, 98 insertions(+), 1 deletion(-)
```

Index: 2.6.23-rc3-mm1/kernel/pid.c

```
=====
--- 2.6.23-rc3-mm1.orig/kernel/pid.c 2007-08-28 22:59:13.000000000 -0700
+++ 2.6.23-rc3-mm1/kernel/pid.c 2007-08-29 22:55:39.000000000 -0700
@@ -199,6 +199,70 @@ static int next_pidmap(struct pid_namesp
    return -1;
}

+/*
+ * Return TRUE if pid namespace @ns1 is an ancestor of pid namespace @ns2.
+ * Return FALSE otherwise (Eg: @ns1 is either a descendant or a distant
+ * cousin of @ns2)
+ */
```

```

+static int ancestor_pid_ns(struct pid_namespace *ns1, struct pid_namespace *ns2)
+{
+ int i;
+ struct pid_namespace *tmp;
+
+ if (ns1 == NULL || ns2 == NULL)
+ return 0;
+
+ if (ns1->level >= ns2->level)
+ return 0;
+
+ tmp = ns2->parent;
+ for (i = tmp->level; i >= ns1->level; i--) {
+ if (tmp == ns1)
+ return 1;
+ tmp = tmp->parent;
+ }
+
+ return 0;
+}
+
+/*
+ * Return TRUE if pid namespace @ns1 is a descendant of pid namespace
+ * of @ns2. Return FALSE otherwise (Eg: @ns1 is either in an ancestor
+ * or a distant cousin of @ns2).
+ */
+static int descendant_pid_ns(struct pid_namespace *ns1,
+ struct pid_namespace *ns2)
+{
+ int i;
+ struct pid_namespace *tmp;
+
+ if (ns1 == NULL || ns2 == NULL)
+ return 0;
+
+ if (ns1->level < ns2->level)
+ return 0;
+
+ tmp = ns1;
+ for (i = tmp->level; i >= ns2->level; i--) {
+ if (tmp == ns2)
+ return 1;
+ tmp = tmp->parent;
+ }
+
+ return 0;
+}
+
+int task_ancestor_pid_ns(struct task_struct *tsk1, struct task_struct *tsk2)

```

```

+{
+ return ancestor_pid_ns(task_active_pid_ns(tsk1),
+     task_active_pid_ns(tsk2));
+}
+
+int task_descendant_pid_ns(struct task_struct *tsk1, struct task_struct *tsk2)
+{
+ return descendant_pid_ns(task_active_pid_ns(tsk1),
+     task_active_pid_ns(tsk2));
+}
+
+fastcall void put_pid(struct pid *pid)
+{
+     struct pid_namespace *ns;

```

Index: 2.6.23-rc3-mm1/include/linux/pid.h

```

=====
--- 2.6.23-rc3-mm1.orig/include/linux/pid.h 2007-08-28 23:00:24.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid.h 2007-08-29 17:36:00.000000000 -0700
@@ -99,6 +99,11 @@ extern void FASTCALL(transfer_pid(struct
 struct pid_namespace;
 extern struct pid_namespace init_pid_ns;

```

```

+extern int task_ancestor_pid_ns(struct task_struct *tsk1,
+ struct task_struct *tsk2);
+extern int task_descendant_pid_ns(struct task_struct *tsk1,
+ struct task_struct *tsk2);
+
+/*
+ * look up a PID in the hash table. Must be called with the tasklist_lock
+ * or rcu_read_lock() held.

```

Index: 2.6.23-rc3-mm1/include/linux/sched.h

```

=====
--- 2.6.23-rc3-mm1.orig/include/linux/sched.h 2007-08-27 20:04:23.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/sched.h 2007-08-28 23:12:54.000000000 -0700
@@ -1290,6 +1290,23 @@ static inline pid_t task_ppid_nr_ns(stru
 return pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
}

```

```

+static inline struct pid_namespace *pid_active_ns(struct pid *pid)
+{
+     if (pid == NULL)
+         return NULL;
+
+     return pid->numbers[pid->level].ns;
+}
+
+/*
+ * Return TRUE if the active pid namespace of @tsk is same as active

```

```
+ * pid namespace of 'current'
+ */
+static inline int pid_ns_equal(struct task_struct *tsk)
+{
+    return pid_active_ns(task_pid(tsk)) == pid_active_ns(task_pid(current));
+}
+
+/**
```

```
 * pid_alive - check that a task structure is not stale
 * @p: Task structure to be checked.
```

Index: 2.6.23-rc3-mm1/include/linux/pid_namespace.h

```
=====
--- 2.6.23-rc3-mm1.orig/include/linux/pid_namespace.h 2007-08-27 20:04:23.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid_namespace.h 2007-08-29 23:00:38.000000000 -0700
@@ -47,7 +47,18 @@ static inline void put_pid_ns(struct pid
```

```
static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
{
- return tsk->nsproxy->pid_ns;
+ /*
+ * If task still has its namespaces (i.e it is not exiting)
+ * get the pid ns from nsproxy.
+ */
+ if (tsk->nsproxy)
+ return tsk->nsproxy->pid_ns;
+
+ /*
+ * If it is exiting but has not been reaped, get pid ns from
+ * pid->numbers[]. Otherwise return NULL.
+ */
+ return pid_active_ns(task_pid(tsk));
}
```

```
static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [RFC][PATCH 1/3] Pid ns helpers for signals

Posted by [Oleg Nesterov](#) on Thu, 30 Aug 2007 08:21:48 GMT

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

On 08/29, sukadev@us.ibm.com wrote:

```
>
> +static int ancestor_pid_ns(struct pid_namespace *ns1, struct pid_namespace *ns2)
> +{
```

```

> + int i;
> + struct pid_namespace *tmp;
> +
> + if (ns1 == NULL || ns2 == NULL)
> + return 0;
> +
> + if (ns1->level >= ns2->level)
> + return 0;

```

Looks like this check is not needed, because

```

> + tmp = ns2->parent;
> + for (i = tmp->level; i >= ns1->level; i--) {
> + if (tmp == ns1)
> + return 1;
> + tmp = tmp->parent;
> + }
> +
> + return 0;
> +}

```

"for ()" does the necessary comparison. The same for descendant_pid_ns().

But do we really need two different functions with 2 arguments? Afaics, we only need is_current_ancestor_pid_ns(tsk).

kill_something_info() needs is_current_ancestor_or_the_same_pid_ns(tsk) which could be trivially implemented using the previous one.

```

> --- 2.6.23-rc3-mm1.orig/include/linux/sched.h 2007-08-27 20:04:23.000000000 -0700
> +++ 2.6.23-rc3-mm1/include/linux/sched.h 2007-08-28 23:12:54.000000000 -0700
> @@ -1290,6 +1290,23 @@ static inline pid_t task_ppid_nr_ns(stru
> return pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
> }
>
> +static inline struct pid_namespace *pid_active_ns(struct pid *pid)
> +{
> + if (pid == NULL)
> + return NULL;
> +
> + return pid->numbers[pid->level].ns;
> +}

```

The function itself is racy, this pid can be already freed. Perhaps not a problem currently, afaics it is only used on signal sending path, the receiver is locked (or the task was found under tasklist), and another task is current.

```
> static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
> {
> - return tsk->nsproxy->pid_ns;
> + /*
> + * If task still has its namespaces (i.e it is not exiting)
> + * get the pid ns from nsproxy.
> + */
> + if (tsk->nsproxy)
> + return tsk->nsproxy->pid_ns;
```

Racy. Needs task_lock() or rcu_lock(). Please see below,

```
> + /*
> + * If it is exiting but has not been reaped, get pid ns from
> + * pid->numbers[]. Otherwise return NULL.
> + */
> + return pid_active_ns(task_pid(tsk));
```

Why can't we just use this chunk and avoid using ->nsproxy?

Oleg.

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