Subject: [PATCH 2/3] Pid ns helpers for signals
Posted by Sukadev Bhattiprolu on Fri, 31 Aug 2007 20:36:34 GMT
View Forum Message <> Reply to Message

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

 is_current_in_ancestor_pid_ns(task)

  TRUE iff active pid namespace of 'current' is an ancestor of
  active pid namespace of @task.

 is_current_in_same_or_ancestor_pid_ns(task)

  TRUE iff active pid namespace of 'current' is either same as
  or an ancestor of active pid namespace of @task.

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

These interfaces take into account the fact that the tasks they are operating
on may be exiting and may have already exited their namespaces (i.e their
pid namespaces may be NULL).

Changelog: [Oleg Nesterov]: Renamed helpers and grab a reference to 'struct
    pid' and 'struct pid_namespace'.
---
 include/linux/pid.h | 15 ++++++
 include/linux/pid_namespace.h | 4 -
 kernel/pid.c | 93 ++
 3 files changed, 110 insertions(+), 2 deletions(-)

Index: 2.6.23-rc3-mm1/include/linux/pid.h
===
--- 2.6.23-rc3-mm1.orig/include/linux/pid.h 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid.h 2007-08-31 12:38:08.000000000 -0700
@@ -125,6 +125,21 @@ extern void FASTCALL(free_pid(struct pid
 extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 /*
+ * Caller must hold a reference to @pid.
+ */
+static inline struct pid_namespace *pid_active_ns(struct pid *pid)
+{
+ if (!pid)
+ return NULL;
+

```
+ return pid->numbers[pid->level].ns;
+}
+
+extern int pid_ns_equal(struct task_struct *tsk);
+extern int is_current_in_ancestor_pid_ns(struct task_struct *tsk);
+extern int is_current_in_same_or_ancestor_pid_ns(struct task_struct *tsk);
+
+/*
  * the helpers to get the pid's id seen from different namespaces
  *
  * pid_nr()    : global id, i.e. the id seen from the init namespace;
Index: 2.6.23-rc3-mm1/kernel/pid.c
===================================================================
--- 2.6.23-rc3-mm1.orig/kernel/pid.c 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/kernel/pid.c 2007-08-31 12:29:10.000000000 -0700
@@ -199,6 +199,99 @@ static int next_pidmap(struct pid_namesp
  return -1;
 }

+static struct pid_namespace *get_task_pid_ns(struct task_struct *tsk)
+{
+ struct pid *pid;
+ struct pid_namespace *ns;
+
+ pid = get_task_pid(tsk, PIDTYPE_PID);
+ ns = get_pid_ns(pid_active_ns(pid));
+ put_pid(pid);
+
+ return ns;
+}
+
+/*
+ * Return TRUE if the active pid namespace of @tsk is same as active
+ * pid namespace of 'current'.
+ */
+int pid_ns_equal(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = (my_ns == tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
```

```
+
+/*
+ * Return TRUE if pid namespace @ns1 is an ancestor of pid namespace @ns2.
+ * Return FALSE otherwise.
+ *
+ * Note: Callers must ensure @ns1 and @ns2 are stable.
+ */
+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 active pid namespace of 'current' is an ancestor of
+ * pid namespace of @tsk. Return FALSE otherwise.
+ */
+int is_current_in_ancestor_pid_ns(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = ancestor_pid_ns(my_ns, tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
+
+/*
+ * Return TRUE if active pid namespace of 'current' is either same as
```

```
+ * or an ancestor of active pid namespace of @tsk.
+ */
+int is_current_in_same_or_ancestor_pid_ns(struct task_struct *tsk)
+{
+ int rc;
+ struct pid_namespace *my_ns = get_task_pid_ns(current);
+ struct pid_namespace *tsk_ns = get_task_pid_ns(tsk);
+
+ rc = my_ns == tsk_ns || ancestor_pid_ns(my_ns, tsk_ns);
+
+ put_pid_ns(my_ns);
+ put_pid_ns(tsk_ns);
+
+ return rc;
+}
+
 fastcall void put_pid(struct pid *pid)
 {
  struct pid_namespace *ns;
Index: 2.6.23-rc3-mm1/include/linux/pid_namespace.h
===================================================================
--- 2.6.23-rc3-mm1.orig/include/linux/pid_namespace.h 2007-08-31 00:45:18.000000000 -0700
+++ 2.6.23-rc3-mm1/include/linux/pid_namespace.h 2007-08-31 12:15:09.000000000 -0700
@@ -31,7 +31,7 @@ extern struct pid_namespace init_pid_ns;

 static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 {
- if (ns != &init_pid_ns)
+ if (ns && ns != &init_pid_ns)
   kref_get(&ns->kref);
  return ns;
 }
@@ -41,7 +41,7 @@ extern void free_pid_ns(struct kref *kre

 static inline void put_pid_ns(struct pid_namespace *ns)
 {
- if (ns != &init_pid_ns)
+ if (ns && ns != &init_pid_ns)
   kref_put(&ns->kref, free_pid_ns);
 }
```