Subject: [PATCH 13/17] Pid-NS(V3) Make pid ns functions inline
Posted by Sukadev Bhattiprolu on Sat, 16 Jun 2007 23:03:19 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 13/17] Pid-NS(V3) Make pid ns functions inline

Now that pid namespace is no longer dependent on 'struct nsproxy' we
can inline is_global_init() and pid_to_nr().

We can also clean up pid_namespace.h by moving task_child_reaper() and
task_active_pid_ns() into sched.h (where other functions that operate
on a 'task_struct' currently reside).

P.S: Note that while these "clean up" changes could be folded into the
    previous patch, they may unnecessarily complicate that patch due to
    dependencies between sched.h, pid.h, nsproxy.h and pid_namespace.h.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 include/linux/pid.h | 5 ----
 include/linux/pid_namespace.h | 37 ---------------------------------
 include/linux/sched.h | 47 ++---
 kernel/nsproxy.c | 1
 kernel/pid.c | 20 ----------------
 5 files changed, 46 insertions(+), 64 deletions(-)

Index: lx26-22-rc4-mm2/include/linux/pid.h
===
--- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 19:21:35.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:51.000000000 -0700
@@ -2,6 +2,7 @@
 #define _LINUX_PID_H

 #include <linux/rcupdate.h>
+#include <linux/pid_namespace.h>

 enum copy_process_type {
  COPY_NON_IDLE_PROCESS,
@@ -16,8 +17,6 @@ enum pid_type
  PIDTYPE_MAX
 };

-struct pid_namespace;
-
 /*
  * A struct upid holds a process identifier (or pid->nr) for a given
  * pid namespace.
@@ -122,7 +121,6 @@ extern struct pid *dup_struct_pid(enum c

```
   unsigned long clone_flags, struct task_struct *new_task);
 extern void FASTCALL(free_pid(struct pid *pid));

-extern pid_t pid_to_nr(struct pid *pid);
 extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 #define do_each_pid_task(pid, type, task)    \
@@ -170,7 +168,6 @@ static inline struct pid_namespace *pid_
  */
 static inline struct pid_namespace *pid_active_pid_ns_parent(struct pid *pid)
 {
- extern struct pid_namespace init_pid_ns;
  struct pid_namespace *ns = &init_pid_ns;

  if (pid->upid_list[0].pid_ns != &init_pid_ns)
Index: lx26-22-rc4-mm2/include/linux/pid_namespace.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/pid_namespace.h 2007-06-15 19:21:35.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid_namespace.h 2007-06-15 19:26:40.000000000 -0700
@@ -1,11 +1,6 @@
 #ifndef _LINUX_PID_NS_H
 #define _LINUX_PID_NS_H

-#include <linux/sched.h>
-#include <linux/mm.h>
-#include <linux/threads.h>
-#include <linux/pid.h>
-#include <linux/nsproxy.h>
 #include <linux/kref.h>

 struct pidmap {
@@ -50,36 +45,4 @@ static inline void put_pid_ns(struct pid
  kref_put(&ns->kref, free_pid_ns);
 }

-extern struct pid_namespace *pid_active_pid_ns(struct pid *pid);
-static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
-{
- return pid_active_pid_ns(task_pid(tsk));
-}
-
-/*
- * Return the child reaper of @tsk.
- *
- * Normally the child reaper of @tsk is simply the child reaper
- * the active pid namespace of @tsk.
- *
- * But if @tsk is itself child reaper of a namespace, NS1, its child
```

```
- * reaper depends on the caller.  If someone from an ancestor namespace
- * or, if the reaper himself is asking, return the reaper of our parent
- * namespace.
- *
- * If someone from namespace NS1 (other than reaper himself) is asking,
- * return reaper of NS1.
- */
-static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
-{
- struct task_struct *reaper;
- struct pid *pid = task_pid(tsk);
-
- reaper = pid_active_pid_ns(pid)->child_reaper;
- if (tsk == reaper)
-  reaper = pid_active_pid_ns_parent(pid)->child_reaper;
-
- return reaper;
-}
-
 #endif /* _LINUX_PID_NS_H */
Index: lx26-22-rc4-mm2/include/linux/sched.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/sched.h 2007-06-15 19:04:53.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/sched.h 2007-06-15 19:47:51.000000000 -0700
@@ -1218,6 +1218,46 @@ static inline struct pid *task_session(s
  return task->group_leader->pids[PIDTYPE_SID].pid;
 }

+static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
+{
+ return pid_active_pid_ns(task_pid(tsk));
+}
+
+/**
+ * Return the child reaper of @tsk.
+ *
+ * Normally the child reaper of @tsk is simply the child reaper
+ * the active pid namespace of @tsk.
+ *
+ * But if @tsk is itself child reaper of a namespace, NS1, its child
+ * reaper depends on the caller.  If someone from an ancestor namespace
+ * or, if the reaper himself is asking, return the reaper of our parent
+ * namespace.
+ *
+ * If someone from namespace NS1 (other than reaper himself) is asking,
+ * return reaper of NS1.
+ */
+static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
```

```
+{
+ struct task_struct *reaper;
+ struct pid *pid = task_pid(tsk);
+
+ reaper = pid_active_pid_ns(pid)->child_reaper;
+ if (tsk == reaper)
+   reaper = pid_active_pid_ns_parent(pid)->child_reaper;
+
+ return reaper;
+}
+
+/**
+ * Return the pid_t by which the process @pid is known in the active
+ * pid namespace of the caller.
+ */
+static inline pid_t pid_to_nr(struct pid *pid)
+{
+ return pid_to_nr_in_ns(pid, task_active_pid_ns(current));
+}
+
 /**
  * pid_alive - check that a task structure is not stale
  * @p: Task structure to be checked.
@@ -1236,10 +1276,11 @@ static inline int pid_alive(struct task_
  * @tsk: Task structure to be checked.
  *
  * Check if a task structure is the first user space task the kernel created.
- *
- * TODO: We should inline this function after some cleanups in pid_namespace.h
  */
-extern int is_global_init(struct task_struct *tsk);
+static inline int is_global_init(struct task_struct *tsk)
+{
+ return tsk == init_pid_ns.child_reaper;
+}

 /*
  * is_container_init:
Index: lx26-22-rc4-mm2/kernel/pid.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/pid.c 2007-06-15 19:21:35.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/pid.c 2007-06-15 19:47:51.000000000 -0700
@@ -111,11 +111,6 @@ struct pid_namespace init_pid_ns = {
  .child_reaper = &init_task
 };

-int is_global_init(struct task_struct *tsk)
-{
```

```
- return tsk == init_pid_ns.child_reaper;
-}
-
 /*
  * Note: disable interrupts while the pidmap_lock is held as an
  * interrupt might come in and do read_lock(&tasklist_lock).
@@ -263,21 +258,6 @@ static int init_upid(struct upid *upid,
  return 0;
 }

-/*
- * Return the pid_t by which the process @pid is known in the active
- * pid namespace of the caller.
- *
- * TODO: pid_to_nr() cannot be easily inlined due to dependencies between
- *      'task_struct', 'nsproxy' and 'pid_namespace'. As a part of
- *      implementing multiple pid namespaces, we remove 'pid_namespace'
- *      from 'nsproxy' and inline this function.
- */
-pid_t pid_to_nr(struct pid *pid)
-{
- return pid_to_nr_in_ns(pid, task_active_pid_ns(current));
-}
-EXPORT_SYMBOL_GPL(pid_to_nr);
-
 #ifdef CONFIG_PID_NS
 static int init_ns_pidmap(struct pid_namespace *ns)
 {
Index: lx26-22-rc4-mm2/kernel/nsproxy.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/nsproxy.c 2007-06-15 19:04:53.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/nsproxy.c 2007-06-15 19:21:35.000000000 -0700
@@ -19,6 +19,7 @@
 #include <linux/init_task.h>
 #include <linux/mnt_namespace.h>
 #include <linux/utsname.h>
+#include <linux/err.h>

 struct nsproxy init_nsproxy = INIT_NSPROXY(init_nsproxy);


--
```