Subject: [RFC][PATCH 16/16] Move inline functions to sched.h
Posted by Sukadev Bhattiprolu on Thu, 24 May 2007 01:15:36 GMT
View Forum Message <> Reply to Message

Subject: Move inline functions to sched.h

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

task_active_pid_ns() and pid_to_nr() cannot both be inline since they
are currently defined in different files (pid_namespace.h and pid.h)
These files depend on each other and sched.h and to resolve the
circular dependency some functions have to be extern even if they all
should really be inline.

This patch moves all these functions to sched.h and makes them all inline.

TODO: Fold these changes into the appropriate patches in the patchset.
Any concern with adding these to sched.h ?

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

```
---
 include/linux/pid.h          |    2
 include/linux/pid_namespace.h |    8 ---
 include/linux/sched.h        |   92 ++++++++++++++++++++++++++++++++++++++++++++
 kernel/pid.c                 |   86 ---------------------------------------
 4 files changed, 92 insertions(+), 96 deletions(-)

Index: lx26-21-mm2/kernel/pid.c
===================================================================
--- lx26-21-mm2.orig/kernel/pid.c 2007-05-23 13:25:56.000000000 -0700
+++ lx26-21-mm2/kernel/pid.c 2007-05-23 13:25:56.000000000 -0700
@@ -246,49 +246,6 @@ static int init_upid(struct upid *upid,
  return 0;
 }

-static struct upid *pid_active_upid(struct pid *pid)
-{
- return &pid->upid_list[0];
-}
-
-/*
- * Return the active pid namespace of the process @pid.
- *
- * Note:
- * To avoid having to use an extra pointer in struct pid to keep track
- * of active pid namespace, dup_struct_pid() maintains the order of
```

```
- * entries in 'pid->upid_list' such that the youngest (or the 'active')
- * pid namespace is the first entry and oldest (init_pid_ns) is the last
- * entry in the list.
- */
-struct pid_namespace *pid_active_pid_ns(struct pid *pid)
-{
- return pid_active_upid(pid)->pid_ns;
-}
-EXPORT_SYMBOL_GPL(pid_active_pid_ns);
-
-/*
- * Return the parent pid_namespace of the active pid namespace of @tsk.
- *
- * Note:
- *  Refer to function header of pid_active_pid_ns() for information on
- *  the order of entries in pid->upid_list. Based on the order, the parent
- *  pid namespace of the active pid namespace of @tsk is just the second
- *  entry in the process's pid->upid_list.
- *
- *  Parent pid namespace of init_pid_ns is init_pid_ns itself.
- */
-static struct pid_namespace *task_active_pid_ns_parent(struct task_struct *tsk)
-{
- int idx = 0;
- struct pid *pid = task_pid(tsk);
-
- if (pid->num_upids > 1)
-  idx++;
-
- return pid->upid_list[idx].pid_ns;
-}
-
 /*
  * Return the child reaper of @tsk.
  *
@@ -325,49 +282,6 @@ struct task_struct *task_child_reaper(st
 }
 EXPORT_SYMBOL(task_child_reaper);

-/*
- * Return the pid_t by which the process @pid is known in the pid
- * namespace @ns.
- *
- * Return 0 if:
- *  - @pid is NULL (eg: procfs calls this for task_pgrp(init_task)
- *    which is NULL).
- *
- *  - process does not have pid_t in the namespace @ns (eg: parent
```

```
- *    process of a child reaper does not exist in the child namespace.
- *    A getppid() call by the child reaper results in 0).
- */
-pid_t pid_to_nr_in_ns(struct pid_namespace *ns, struct pid *pid)
-{
- int i;
- struct upid *upid;
-
- if (!pid)
-  return 0;
-
- upid = &pid->upid_list[0];
- for (i = 0; i < pid->num_upids; i++, upid++) {
-  if (upid->pid_ns == ns)
-   return upid->nr;
- }
- return 0;
-}
-EXPORT_SYMBOL_GPL(pid_to_nr_in_ns);
-
-/*
- * Return the pid_t by which the process @pid is known in the active
- * pid namespace of the caller.
- *
- * pid_to_nr() cannot be static inline if task_active_pid_ns() is
- * inline as it would cause a circular dependency between pid.h
- * and pid_namespace.h.
- */
-pid_t pid_to_nr(struct pid *pid)
-{
- return pid_to_nr_in_ns(task_active_pid_ns(current), pid);
-}
-EXPORT_SYMBOL_GPL(pid_to_nr);
-
 #ifdef CONFIG_PID_NS
 static int init_ns_pidmap(struct pid_namespace *ns)
 {
Index: lx26-21-mm2/include/linux/sched.h
===================================================================
--- lx26-21-mm2.orig/include/linux/sched.h 2007-05-23 13:25:56.000000000 -0700
+++ lx26-21-mm2/include/linux/sched.h 2007-05-23 13:25:56.000000000 -0700
@@ -1176,6 +1176,98 @@ struct pid_namespace;
 extern int is_global_init(struct task_struct *tsk);
 extern int is_container_init(struct task_struct *tsk);

+/*
+ * Return the active upid of the process @pid.
+ *
```

```
+ * Note:
+ * To avoid having to use an extra pointer in struct pid to keep track
+ * of active pid namespace, dup_struct_pid() maintains the order of
+ * entries in 'pid->upid_list' such that the youngest (or the 'active')
+ * pid namespace is the first entry and oldest (init_pid_ns) is the last
+ * entry in the list.
+ */
+static inline struct upid *pid_active_upid(struct pid *pid)
+{
+ return &pid->upid_list[0];
+}
+
+static inline struct pid_namespace *pid_active_pid_ns(struct pid *pid)
+{
+ return pid_active_upid(pid)->pid_ns;
+}
+
+static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
+{
+ return pid_active_pid_ns(task_pid(tsk));
+}
+
+/*
+ * Return the parent pid_namespace of the active pid namespace of @tsk.
+ *
+ * Note:
+ *  Refer to function header of pid_active_pid_ns() for information on
+ *  the order of entries in pid->upid_list. Based on the order, the parent
+ *  pid namespace of the active pid namespace of @tsk is just the second
+ *  entry in the process's pid->upid_list.
+ *
+ *  Parent pid namespace of init_pid_ns is init_pid_ns itself.
+ */
+static inline struct pid_namespace *task_active_pid_ns_parent(
+  struct task_struct *tsk)
+{
+ int idx = 0;
+ struct pid *pid = task_pid(tsk);
+
+ if (pid->num_upids > 1)
+  idx++;
+
+ return pid->upid_list[idx].pid_ns;
+}
+
+extern struct task_struct *task_child_reaper(struct task_struct *tsk);
+
+/*
```

```
+ * Return the pid_t by which the process @pid is known in the pid
+ * namespace @ns.
+ *
+ * Return 0 if:
+ *  - @pid is NULL (eg: procfs calls this for task_pgrp(init_task)
+ *    which is NULL).
+ *
+ *  - process does not have pid_t in the namespace @ns (eg: parent
+ *    process of a child reaper does not exist in the child namespace.
+ *    A getppid() call by the child reaper results in 0).
+ */
+static inline pid_t pid_to_nr_in_ns(struct pid_namespace *ns, struct pid *pid)
+{
+ int i;
+ struct upid *upid;
+
+ WARN_ON(pid == 0);
+ if (!pid)
+  return 0;
+
+ upid = &pid->upid_list[0];
+ for (i = 0; i < pid->num_upids; i++, upid++) {
+  if (upid->pid_ns == ns)
+   return upid->nr;
+ }
+ return 0;
+}
+
+/*
+ * Return the pid_t by which the process @pid is known in the active
+ * pid namespace of the caller.
+ *
+ * pid_to_nr() cannot be static inline if task_active_pid_ns() is
+ * inline as it would cause a circular dependency between pid.h
+ * and pid_namespace.h.
+ */
+static inline pid_t pid_to_nr(struct pid *pid)
+{
+ return pid_to_nr_in_ns(task_active_pid_ns(current), pid);
+}
+
 extern struct pid *cad_pid;

 extern void free_task(struct task_struct *tsk);
Index: lx26-21-mm2/include/linux/pid.h
===================================================================
--- lx26-21-mm2.orig/include/linux/pid.h 2007-05-23 13:25:56.000000000 -0700
+++ lx26-21-mm2/include/linux/pid.h 2007-05-23 13:25:56.000000000 -0700
```

```
@@ -122,8 +122,6 @@ 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_in_ns(struct pid_namespace *ns, 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)    \
Index: lx26-21-mm2/include/linux/pid_namespace.h
===================================================================
--- lx26-21-mm2.orig/include/linux/pid_namespace.h 2007-05-23 13:25:56.000000000 -0700
+++ lx26-21-mm2/include/linux/pid_namespace.h 2007-05-23 13:25:56.000000000 -0700
@@ -44,18 +44,10 @@ static inline void get_pid_ns(struct pid
 }

 extern void free_pid_ns(struct kref *kref);
-extern struct pid_namespace *pid_active_pid_ns(struct pid *pid);

 static inline void put_pid_ns(struct pid_namespace *ns)
 {
  kref_put(&ns->kref, free_pid_ns);
 }

-static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
-{
- return pid_active_pid_ns(task_pid(tsk));
-}
-
-extern struct task_struct *task_child_reaper(struct task_struct *tsk);
-
 #endif /* _LINUX_PID_NS_H */
```

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