Subject: [RFC][PATCH 04/14] Add struct pid_nr and pid_nrs list
Posted by Sukadev Bhattiprolu on Wed, 21 Mar 2007 03:19:21 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 04/14] Add struct pid_nr and pid_nrs list

A struct pid_nr associates a pid_t value with a pid namespace. We attach
a list of struct pid_nr entries to a struct pid (pid->pid_nrs), allowing
the struct pid to take different pid_t values in different namespaces.

The pid->pid_nrs list and the helper functions will be used in subsequent
functions.

Changelog:
 - [Serge Hallyn comment]: Remove (!pid_nr) check in free_pid_nr()

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
---
 include/linux/pid.h |   23 ++++++++++++++++++++++
 kernel/pid.c        |   49 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 72 insertions(+)

Index: lx26-21-rc3-mm2/include/linux/pid.h
===================================================================
--- lx26-21-rc3-mm2.orig/include/linux/pid.h 2007-03-20 15:55:52.000000000 -0700
+++ lx26-21-rc3-mm2/include/linux/pid.h 2007-03-20 17:18:10.000000000 -0700
@@ -11,6 +11,25 @@ enum pid_type
  PIDTYPE_MAX
 };

+struct pid_namespace;
+
+/*
+ * A struct pid_nr holds a process identifier (or pid->nr) for a given
+ * pid namespace.
+ *
+ * A list of struct pid_nr is stored in the struct pid. this list is
+ * used to get the process identifier associated with the pid
+ * namespace it is being seen from.
+ */
+struct pid_nr
+{
+ struct hlist_node node;
+ int nr;
+ struct hlist_node pid_chain;
+ struct pid *pid;
```

```
+ struct pid_namespace* pid_ns;
+};
+
 /*
  * What is struct pid?
  *
@@ -49,6 +68,7 @@ struct pid
  /* lists of tasks that use this pid */
  struct hlist_head tasks[PIDTYPE_MAX];
  struct rcu_head rcu;
+ struct hlist_head pid_nrs;
 };

 extern struct pid init_struct_pid;
@@ -95,6 +115,9 @@ extern struct pid *FASTCALL(find_pid(int
 extern struct pid *find_get_pid(int nr);
 extern struct pid *find_ge_pid(int nr);

+extern void free_pid_nr(struct pid_nr *pid_nr);
+extern struct pid_nr *alloc_pid_nr(struct pid_namespace *pid_ns,
+    struct pid *pid);
 extern struct pid *alloc_pid(void);
 extern void FASTCALL(free_pid(struct pid *pid));

Index: lx26-21-rc3-mm2/kernel/pid.c
===================================================================
--- lx26-21-rc3-mm2.orig/kernel/pid.c 2007-03-20 15:55:52.000000000 -0700
+++ lx26-21-rc3-mm2/kernel/pid.c 2007-03-20 17:53:54.000000000 -0700
@@ -33,6 +33,7 @@
 static struct hlist_head *pid_hash;
 static int pidhash_shift;
 static struct kmem_cache *pid_cachep;
+static struct kmem_cache *pid_nr_cachep;
 struct pid init_struct_pid = INIT_STRUCT_PID;

 int pid_max = PID_MAX_DEFAULT;
@@ -190,6 +191,16 @@ static void delayed_put_pid(struct rcu_h
  put_pid(pid);
 }

+void free_pid_nr(struct pid_nr *pid_nr)
+{
+ free_pidmap(pid_nr->pid_ns, pid_nr->nr);
+
+ hlist_del_init(&pid_nr->node);
+
+ put_pid_ns(pid_nr->pid_ns);
+ kmem_cache_free(pid_nr_cachep, pid_nr);
```

```
+}
+
 fastcall void free_pid(struct pid *pid)
 {
  /* We can be called with write_lock_irq(&tasklist_lock) held */
@@ -203,6 +214,40 @@ fastcall void free_pid(struct pid *pid)
  call_rcu(&pid->rcu, delayed_put_pid);
 }

+struct pid_nr *alloc_pid_nr(struct pid_namespace *pid_ns, struct pid *pid)
+{
+ struct pid_nr* pid_nr;
+ int nr;
+
+ nr = alloc_pidmap(pid_ns);
+ if (nr < 0)
+  return NULL;
+
+ pid_nr = kmem_cache_alloc(pid_nr_cachep, GFP_KERNEL);
+ if (!pid_nr) {
+  free_pidmap(pid_ns, nr);
+  return NULL;
+ }
+
+ INIT_HLIST_NODE(&pid_nr->node);
+ INIT_HLIST_NODE(&pid_nr->pid_chain);
+ pid_nr->pid_ns = pid_ns;
+ get_pid_ns(pid_ns);
+ pid_nr->nr = nr;
+ /*
+  * The struct pid and list of struct pid_nrs represent a process
+  * with multiple pid_t values, one in each pid namespace. The list
+  * of pid_t values of a process, represented by pid->pid_nrs list,
+  * never changes during the life of the process. As such, struct
+  * pid and its pid_nrs list maybe viewed as a single object i.e
+  * they are created/destroyed together.  So we do not need a
+  * reference to struct pid here.
+  */
+ pid_nr->pid = pid;
+
+ return pid_nr;
+}
+
 struct pid *alloc_pid(void)
 {
  struct pid *pid;
@@ -415,4 +460,8 @@ void __init pidmap_init(void)
  pid_cachep = kmem_cache_create("pid", sizeof(struct pid),
```

```
     __alignof__(struct pid),
     SLAB_PANIC, NULL, NULL);
+
+ pid_nr_cachep = kmem_cache_create("pid_nr", sizeof(struct pid_nr),
+       __alignof__(struct pid_nr),
+       SLAB_PANIC, NULL, NULL);
 }
```

_____

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