Subject: [PATCH 4/13] Introduce the vpid fields and helpers for getting them
Posted by Pavel Emelianov on Thu, 24 May 2007 12:44:07 GMT
View Forum Message <> Reply to Message

Since we want to export virtual pid to userspace and this is
optional (CONFIG_PID_NS) we need helpers for getting the values
of vpid/vtgid/etc depending on the config and the appropriate
members on structs.

A note about the struct pid. As will be seen later pid may now
be stored in two hashes - pid_hash and the vpid_hash. The latter
is used to find the struct pid by pid get from the userspace.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/include/linux/pid.h b/include/linux/pid.h
index 1e0e4e3..3a30f8a 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -46,6 +46,11 @@ struct pid
  /* Try to keep pid_chain in the same cacheline as nr for find_pid */
  int nr;
  struct hlist_node pid_chain;
+#ifdef CONFIG_PID_NS
+ int vnr;
+ struct hlist_node vpid_chain;
+ struct pid_namespace *ns;
+#endif
  /* lists of tasks that use this pid */
  struct hlist_head tasks[PIDTYPE_MAX];
  struct rcu_head rcu;
@@ -106,6 +114,20 @@ static inline pid_t pid_nr(struct pid *p
  return nr;
 }

+#ifdef CONFIG_PID_NS
+static inline pid_t pid_vnr(struct pid *pid)
+{
+ pid_t nr = 0;
+ if (pid)
+  nr = pid->vnr;
+ return nr;
+}
+#else
+#define pid_vnr(pid) pid_nr(pid)
+#endif
```

```
+
+#define pid_nr_ns(pid, ns) (ns == &init_pid_ns ? pid_nr(pid) : pid_vnr(pid))
+
 #define do_each_pid_task(pid, type, task)    \
  do {        \
   struct hlist_node *pos___;    \
diff --git a/include/linux/sched.h b/include/linux/sched.h
index d4de6d8..7743a11 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -456,7 +457,10 @@ struct signal_struct {
   pid_t session __deprecated;
   pid_t __session;
  };
-
+#ifdef CONFIG_PID_NS
+ pid_t vpgrp;
+ pid_t vsession;
+#endif
  /* boolean value for session group leader */
  int leader;

@@ -887,6 +891,10 @@ struct task_struct {
  unsigned did_exec:1;
  pid_t pid;
  pid_t tgid;
+#ifdef CONFIG_PID_NS
+ pid_t vpid;
+ pid_t vtgid;
+#endif

 #ifdef CONFIG_CC_STACKPROTECTOR
  /* Canary value for the -fstack-protector gcc feature */
@@ -1127,6 +1135,128 @@ static inline struct pid *task_session(s
  return task->group_leader->pids[PIDTYPE_SID].pid;
 }

+#ifdef CONFIG_PID_NS
+static inline pid_t task_pid_vnr(struct task_struct *tsk)
+{
+ return tsk->vpid;
+}
+
+static inline void set_task_vpid(struct task_struct *tsk, pid_t nr)
+{
+ tsk->vpid = nr;
+}
+
```

```
+static inline pid_t task_tgid_vnr(struct task_struct *tsk)
+{
+ return tsk->vtgid;
+}
+
+static inline void set_task_vtgid(struct task_struct *tsk, pid_t nr)
+{
+ tsk->vtgid = nr;
+}
+
+static inline pid_t task_session_vnr(struct task_struct *tsk)
+{
+ return tsk->signal->vsession;
+}
+
+static inline void set_task_vsession(struct task_struct *tsk, pid_t nr)
+{
+ tsk->signal->vsession = nr;
+}
+
+static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
+{
+ return tsk->signal->vpgrp;
+}
+
+static inline void set_task_vpgrp(struct task_struct *tsk, pid_t nr)
+{
+ tsk->signal->vpgrp = nr;
+}
+
+extern struct pid_namespace init_pid_ns;
+
+static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ if (ns == &init_pid_ns)
+  return rcu_dereference(tsk->real_parent)->tgid;
+
+ if (tsk->vpid == 1)
+  return 0;
+
+ return rcu_dereference(tsk->real_parent)->vtgid;
+}
+#else
+static inline pid_t task_pid_vnr(struct task_struct *tsk)
+{
+ return tsk->pid;
+}
```

```
+
+static inline void set_task_vpid(struct task_struct *tsk, pid_t nr)
+{
+}
+
+static inline pid_t task_tgid_vnr(struct task_struct *tsk)
+{
+ return tsk->tgid;
+}
+
+static inline void set_task_vtgid(struct task_struct *tsk, pid_t nr)
+{
+}
+
+static inline pid_t task_session_vnr(struct task_struct *tsk)
+{
+ return task_session_nr(tsk);
+}
+
+static inline void set_task_vsession(struct task_struct *tsk, pid_t nr)
+{
+}
+
+static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
+{
+ return task_pgrp_nr(tsk);
+}
+
+static inline void set_task_vpgrp(struct task_struct *tsk, pid_t nr)
+{
+}
+
+static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return rcu_dereference(tsk->real_parent)->tgid;
+}
+#endif
+
+#define __task_pid_nr_ns(tsk, ns) \
+ (ns == &init_pid_ns ? tsk->pid : task_pid_vnr(tsk))
+
+#define task_pid_nr_ns(tsk)  \
+ __task_pid_nr_ns(tsk, current->nsproxy->pid_ns)
+
+#define __task_tgid_nr_ns(tsk, ns) \
+ (ns == &init_pid_ns ? tsk->tgid : task_tgid_vnr(tsk))
+
```

```
+#define task_tgid_nr_ns(tsk)  \
+ __task_tgid_nr_ns(tsk, current->nsproxy->pid_ns)
+
+#define __task_pgrp_nr_ns(tsk, ns) \
+ (ns == &init_pid_ns ? task_pgrp_nr(tsk) : task_pgrp_vnr(tsk))
+
+#define task_pgrp_nr_ns(tsk)  \
+ __task_pgrp_nr_ns(tsk, current->nsproxy->pid_ns)
+
+#define __task_session_nr_ns(tsk, ns) \
+ (ns == &init_pid_ns ? task_session_nr(tsk) : task_session_vnr(tsk))
+
+#define task_session_nr_ns(tsk)  \
+ __task_session_nr_ns(tsk, current->nsproxy->pid_ns)
+
 /**
  * pid_alive - check that a task structure is not stale
  * @p: Task structure to be checked.
```