Subject: Re: [PATCH 6/16] Helpers to obtain pid numbers
Posted by Sukadev Bhattiprolu on Tue, 10 Jul 2007 05:18:01 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| When showing pid to user or getting the pid numerical id for in-kernel
| use the value of this id may differ depending on the namespace.
|
| This set of helpers is used to get the global pid nr, the virtual (i.e.
| seen by task in its namespace) nr and the nr as it is seen from the
| specified namespace.
|
| Signed-off-by: Pavel Emelianov <xemul@openvz.org>
|
| ---
|
| include/linux/pid.h | 27 ++++++++++++
| include/linux/sched.h | 108 +++-----
| kernel/pid.c | 8 +++
| 3 files changed, 132 insertions(+), 11 deletions(-)
|
| diff -upr linux-2.6.22-rc4-mm2.orig/include/linux/pid.h linux-2.6.22-rc4-mm2-2/include/linux/pid.h
| --- linux-2.6.22-rc4-mm2.orig/include/linux/pid.h 2007-06-14 12:14:29.000000000 +0400
| +++ linux-2.6.22-rc4-mm2-2/include/linux/pid.h 2007-07-04 19:00:38.000000000 +0400
| @@ -83,6 +89,9 @@ extern void FASTCALL(detach_pid(struct t
| extern void FASTCALL(transfer_pid(struct task_struct *old,
| struct task_struct *new, enum pid_type));
|
| +struct pid_namespace;
| +extern struct pid_namespace init_pid_ns;
| +
| /*
| * look up a PID in the hash table. Must be called with the tasklist_lock
| * or rcu_read_lock() held.
| @@ -93,14 +99,36 @@ extern void FASTCALL(detach_pid(struct t
| extern struct pid *alloc_pid(void);
| extern void FASTCALL(free_pid(struct pid *pid));
|
| +/*
| + * 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;
| + * pid_vnr()   : virtual id, i.e. the id seen from the namespace this pid
| + *   belongs to. this only makes sence when called in the
| + *   context of the task that belongs to the same namespace;
| + * pid_nr_ns() : id seen from the ns specified.
| + *
| + * see also task_xid_nr() etc in include/linux/sched.h

```
| + */
```

I think its a bit confusing and error-prone to have both pid_nr() and pid_vnr().

BTW, shouldn't you use pid_vnr() in do_task_stat() ? You currently use pid_nr()
and that returns the init-pid-ns id right ?

```
| +
|  static inline pid_t pid_nr(struct pid *pid)
|  {
|   pid_t nr = 0;
|   if (pid)
| -  nr = pid->nr;
| +  nr = pid->numbers[0].nr;
|   return nr;
|  }
|
| +pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns);
| +
| +static inline pid_t pid_vnr(struct pid *pid)
| +{
| + pid_t nr = 0;
| + if (pid)
| +  nr = pid->numbers[pid->level].nr;
| + return nr;
| +}
| +
|  #define do_each_pid_task(pid, type, task)    \
|   do {        \
|    struct hlist_node *pos___;    \
| diff -upr linux-2.6.22-rc4-mm2.orig/kernel/pid.c linux-2.6.22-rc4-mm2-2/kernel/pid.c
| --- linux-2.6.22-rc4-mm2.orig/kernel/pid.c 2007-06-14 12:14:29.000000000 +0400
| +++ linux-2.6.22-rc4-mm2-2/kernel/pid.c 2007-07-04 19:00:38.000000000 +0400
| @@ -339,6 +379,14 @@ struct pid *find_get_pid(pid_t nr)
|   return pid;
|  }
|
| +pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
| +{
| + pid_t nr = 0;
| + if (pid && ns->level <= pid->level)
| +  nr = pid->numbers[ns->level].nr;
| + return nr;
| +}
| +
|  /*
|   * Used by proc to find the first pid that is greater then or equal to nr.
```

```
|   *
| diff -upr linux-2.6.22-rc4-mm2.orig/include/linux/sched.h
linux-2.6.22-rc4-mm2-2/include/linux/sched.h
| --- linux-2.6.22-rc4-mm2.orig/include/linux/sched.h 2007-07-04 19:00:38.000000000 +0400
| +++ linux-2.6.22-rc4-mm2-2/include/linux/sched.h 2007-07-04 19:00:38.000000000 +0400
| @@ -1153,16 +1154,6 @@ struct task_struct {
|  #endif
|  };
|
| -static inline pid_t task_pgrp_nr(struct task_struct *tsk)
| -{
| - return tsk->signal->pgrp;
| -}
| -
| -static inline pid_t task_session_nr(struct task_struct *tsk)
| -{
| - return tsk->signal->__session;
| -}
| -
|  static inline void set_task_session(struct task_struct *tsk, pid_t session)
|  {
|   tsk->signal->__session = session;
| @@ -1188,6 +1179,104 @@ static inline struct pid *task_session(s
|   return task->group_leader->pids[PIDTYPE_SID].pid;
|  }
|
| +struct pid_namespace;
| +
| +/*
| + * the helpers to get the task's different pids as they are seen
| + * from various namespaces
| + *
| + * task_xid_nr()     : global id, i.e. the id seen from the init namespace;
| + * task_xid_vnr()    : virtual id, i.e. the id seen from the namespace the task
| + *                     belongs to. this only makes sence when called in the
| + *                     context of the task that belongs to the same namespace;
| + * task_xid_nr_ns()  : id seen from the ns specified;
| + *
| + * set_task_vxid()   : assigns a virtual id to a task;
| + *
| + * task_ppid_nr_ns() : the parent's id as seen from the namespace specified.
| + *                     the result depends on the namespace and whether the
| + *                     task in question is the namespace's init. e.g. for the
| + *                     namespace's init this will return 0 when called from
| + *                     the namespace of this init, or appropriate id otherwise.
| + *
| + *
| + * see also pid_nr() etc in include/linux/pid.h
```

```
| + */
| +
| +static inline pid_t task_pid_nr(struct task_struct *tsk)
| +{
| + return tsk->pid;
| +}
| +
| +static inline pid_t task_pid_nr_ns(struct task_struct *tsk,
| +  struct pid_namespace *ns)
| +{
| + return pid_nr_ns(task_pid(tsk), ns);
| +}
| +
| +static inline pid_t task_pid_vnr(struct task_struct *tsk)
| +{
| + return pid_vnr(task_pid(tsk));
| +}
| +
| +
| +static inline pid_t task_tgid_nr(struct task_struct *tsk)
| +{
| + return tsk->tgid;
| +}
| +
| +static inline pid_t task_tgid_nr_ns(struct task_struct *tsk,
| +  struct pid_namespace *ns)
| +{
| + return pid_nr_ns(task_tgid(tsk), ns);
| +}
| +
| +static inline pid_t task_tgid_vnr(struct task_struct *tsk)
| +{
| + return pid_vnr(task_tgid(tsk));
| +}
| +
| +
| +static inline pid_t task_pgrp_nr(struct task_struct *tsk)
| +{
| + return tsk->signal->pgrp;
| +}
| +
| +static inline pid_t task_pgrp_nr_ns(struct task_struct *tsk,
| +  struct pid_namespace *ns)
| +{
| + return pid_nr_ns(task_pgrp(tsk), ns);
| +}
| +
| +static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
```

```
| +{
| + return pid_vnr(task_pgrp(tsk));
| +}
| +
| +
| +static inline pid_t task_session_nr(struct task_struct *tsk)
| +{
| + return tsk->signal->__session;
| +}
| +
| +static inline pid_t task_session_nr_ns(struct task_struct *tsk,
| +  struct pid_namespace *ns)
| +{
| + return pid_nr_ns(task_session(tsk), ns);
| +}
| +
| +static inline pid_t task_session_vnr(struct task_struct *tsk)
| +{
| + return pid_vnr(task_session(tsk));
| +}
| +
| +
| +static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
| +  struct pid_namespace *ns)
| +{
| + return pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
| +}
| +
|  /**
|   * pid_alive - check that a task structure is not stale
|   * @p: Task structure to be checked.
```

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