

---

Subject: Re: [RFC][PATCH 1/6] Add struct pid\_nr  
Posted by [ebiederm](#) on Sun, 11 Mar 2007 11:35:16 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com writes:

```
> From: Cedric Le Goater <clg@fr.ibm.com>
> Subject: [RFC][PATCH 1/6] Add struct pid_nr
>
> Define struct pid_nr and some helper functions that will be used in
> subsequent patches.
>
> 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 | 21 +++++
> kernel/pid.c | 64 +++++
> 2 files changed, 85 insertions(+)
>
> Index: lx26-20-mm2b/include/linux/pid.h
> =====
> --- lx26-20-mm2b.orig/include/linux/pid.h 2007-03-09 18:13:04.000000000 -0800
> +++ lx26-20-mm2b/include/linux/pid.h 2007-03-09 18:27:51.000000000 -0800
> @@ -11,6 +11,23 @@ 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 pid_namespace* pid_ns;
> +};
> +
```

```

> /*
>  * What is struct pid?
>  *
>  @@ -72,6 +89,7 @@ extern struct task_struct *FASTCALL(get_
>      enum pid_type));
>
> extern struct pid *get_task_pid(struct task_struct *task, enum pid_type type);
> +extern void free_pidmap_pid_nr(struct pid_nr *);
>
> /*
>  * attach_pid() and detach_pid() must be called with the tasklist_lock
>  @@ -95,6 +113,9 @@ extern struct pid *FASTCALL(find_pid(int
> extern struct pid *find_get_pid(int nr);
> extern struct pid *find_ge_pid(int nr);
>
> +extern int attach_pid_nr(struct pid *pid, struct pid_nr *pid_nr);
> +extern void free_pid_nr(struct pid_nr *pid_nr);
> +extern struct pid_nr *alloc_pid_nr(struct pid_namespace *pid_ns);
> extern struct pid *alloc_pid(void);
> extern void FASTCALL(free_pid(struct pid *pid));
>
> Index: lx26-20-mm2b/kernel/pid.c
> =====
> --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 18:13:04.000000000 -0800
> +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 18:27:51.000000000 -0800
> @@ -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 +191,12 @@ static void delayed_put_pid(struct rcu_h
> put_pid(pid);
> }
>
> +void free_pid_nr(struct pid_nr *pid_nr)
> +{
> + 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 +210,59 @@ 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_nr* pid_nr;
> +
> + pid_nr = kmem_cache_alloc(pid_nr_cachep, GFP_KERNEL);
> + if (!pid_nr)
> + return pid_nr;
> +
> + INIT_HLIST_NODE(&pid_nr->node);
> + pid_nr->nr = -1;
> + get_pid_ns(pid_ns);
> + pid_nr->pid_ns = pid_ns;
> +
> + return pid_nr;
> +}
> +
> +void free_pidmap_pid_nr(struct pid_nr *pid_nr)
> +{
> +     free_pidmap(pid_nr->pid_ns, pid_nr->nr);
> +     free_pid_nr(pid_nr);
> +}
> +
> +static struct pid_nr *alloc_pidmap_pid_nr(struct pid_namespace *pid_ns)
> +{
> +     int nr;
> +     struct pid_nr *pid_nr;
> +
> +     nr = alloc_pidmap(pid_ns);
> +     if (nr < 0)
> +         return NULL;
> +
> +     pid_nr = alloc_pid_nr(pid_ns);
> +     if (!pid_nr)
> +         goto out_free_pidmap;
> +
> +     pid_nr->nr = nr;
> +
> +     return pid_nr;
> +
> +out_free_pidmap:
> +     free_pidmap(pid_ns, nr);
> +
> +     return NULL;
> +}
> +
> +int attach_pid_nr(struct pid *pid, struct pid_nr *pid_nr)

```

```
> +{  
> + spin_lock(&pid->lock);  
> + hlist_add_head_rcu(&pid_nr->node, &pid->pid_nrs);  
> + spin_unlock(&pid->lock);
```

struct pid doesn't have a lock member.

We should be able to add everything to struct pid at allocation time, so we should not need a lock.

If you made struct pid\_nr what the hash table entry it would probably make more sense, and gave it a struct pid pointer it would probably make more sense.

Eric

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---