

---

Subject: [PATCH 6/13] Pid allocation/freeing procedures  
Posted by [Pavel Emelianov](#) on Thu, 24 May 2007 12:49:47 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

This patch make alloc\_pid() and free\_pid() aware of the namespaces. When a pid is created not in init namespace it gets into two hashes and holds the pointer to the namespace itself.

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

---

```
diff --git a/kernel/pid.c b/kernel/pid.c
index eb66bd2..1815af4 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -54,6 +54,12 @@ static inline int mk_pid(struct pid_name
#define find_next_offset(map, off) \
    find_next_zero_bit((map)->page, BITS_PER_PAGE, off)

#ifdef CONFIG_PID_NS
+static struct hlist_head *vpid_hash;
+#define vpid_hashfn(nr, ns) hash_long((unsigned long)nr + (unsigned long)ns, \
+ pidhash_shift)
+#endif
+
+/*
+ * PID-map pages start out as NULL, they get allocated upon
+ * first use and are never deallocated. This way a low pid_max
@@ -197,9 +203,19 @@ fastcall void free_pid(struct pid *pid)

    spin_lock_irqsave(&pidmap_lock, flags);
    hlist_del_rcu(&pid->pid_chain);
#ifdef CONFIG_PID_NS
+ if (pid->ns != &init_pid_ns)
+ hlist_del_rcu(&pid->vpid_chain);
+#endif
    spin_unlock_irqrestore(&pidmap_lock, flags);

    free_pidmap(&init_pid_ns, pid->nr);
#ifdef CONFIG_PID_NS
+ if (pid->ns != &init_pid_ns) {
+ free_pidmap(pid->ns, pid->vnr);
+ put_pid_ns(pid->ns);
+ }
+#endif
    call_rcu(&pid->rcu, delayed_put_pid);
```

```
}
```

```
@@ -207,28 +223,52 @@ struct pid *alloc_pid(void)
```

```
{  
    struct pid *pid;  
    enum pid_type type;  
- int nr = -1;  
+ int nr, vnr;  
+ struct pid_namespace *ns;  
  
    pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);  
    if (!pid)  
        goto out;  
  
- nr = alloc_pidmap(current->nsproxy->pid_ns);  
+ vnr = nr = alloc_pidmap(&init_pid_ns);  
    if (nr < 0)  
        goto out_free;  
  
+ ns = current->nsproxy->pid_ns;  
+#ifdef CONFIG_PID_NS  
+ if (ns != &init_pid_ns) {  
+     vnr = alloc_pidmap(ns);  
+     if (vnr < 0)  
+         goto out_free_map;  
+  
+     get_pid_ns(ns);  
+ }  
+#endif  
    atomic_set(&pid->count, 1);  
    pid->nr = nr;  
+#ifdef CONFIG_PID_NS  
+ pid->vnr = vnr;  
+ pid->ns = ns;  
+#endif  
    for (type = 0; type < PIDTYPE_MAX; ++type)  
        INIT_HLIST_HEAD(&pid->tasks[type]);  
  
    spin_lock_irq(&pidmap_lock);  
    hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr)]);  
+#ifdef CONFIG_PID_NS  
+ if (ns != &init_pid_ns)  
+     hlist_add_head_rcu(&pid->vpid_chain,  
+         &vpid_hash[vpid_hashfn(vnr, ns)]);  
+#endif  
    spin_unlock_irq(&pidmap_lock);  
  
out:
```

```

return pid;

#ifdef CONFIG_PID_NS
+out_free_map:
+ free_pidmap(&init_pid_ns, nr);
#endif
out_free:
    kmem_cache_free(pid_cachep, pid);
    pid = NULL;
@@ -397,12 +607,17 @@ void __init pidhash_init(void)
    printk("PID hash table entries: %d (order: %d, %Zd bytes)\n",
        pidhash_size, pidhash_shift,
        pidhash_size * sizeof(struct hlist_head));
-
#ifdef CONFIG_PID_NS
+ pidhash_size *= 2;
#endif
    pid_hash = alloc_bootmem(pidhash_size * sizeof(*(pid_hash)));
    if (!pid_hash)
        panic("Could not alloc pidhash!\n");
    for (i = 0; i < pidhash_size; i++)
        INIT_HLIST_HEAD(&pid_hash[i]);
#ifdef CONFIG_PID_NS
+ vpid_hash = pid_hash + (pidhash_size / 2);
#endif
}

void __init pidmap_init(void)

```

---