
Subject: [RFC][PATCH 3/7] VPIDs: fork modifications

Posted by [dev](#) on Thu, 02 Feb 2006 16:24:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch changes forking procedure. Basically it adds a function `do_fork_pid()` that allows forking task with predefined virtual pid. Also it adds required vpid manipulations on `fork()`. Simple and straightforward.

```
--- ./kernel/fork.c.vpid_fork 2006-02-02 14:33:58.811003152 +0300
```

```
+++ ./kernel/fork.c 2006-02-02 14:41:40.806769120 +0300
```

```
@@ -871,7 +871,7 @@ static task_t *copy_process(unsigned lon
```

```
    unsigned long stack_size,
```

```
    int __user *parent_tidptr,
```

```
    int __user *child_tidptr,
```

```
    int pid)
```

```
+    int pid, int vpid)
```

```
{
```

```
    int retval;
```

```
    struct task_struct *p = NULL;
```

```
@@ -932,9 +932,11 @@ static task_t *copy_process(unsigned lon
```

```
    p->did_exec = 0;
```

```
    copy_flags(clone_flags, p);
```

```
    p->pid = pid;
```

```
+ if (set_virt_pid(p, alloc_vpid(p->pid, vpid ? : -1)) < 0)
```

```
+ goto bad_fork_cleanup_module;
```

```
    retval = -EFAULT;
```

```
    if (clone_flags & CLONE_PARENT_SETTID)
```

```
- if (put_user(p->pid, parent_tidptr))
```

```
+ if (put_user(virt_pid(p), parent_tidptr))
```

```
    goto bad_fork_cleanup;
```

```
    p->proc_dentry = NULL;
```

```
@@ -985,8 +987,13 @@ static task_t *copy_process(unsigned lon
```

```
#endif
```

```
    p->tgid = p->pid;
```

```
- if (clone_flags & CLONE_THREAD)
```

```
+ set_virt_tgid(p, virt_pid(p));
```

```
+ set_virt_pgid(p, virt_pgid(current));
```

```
+ set_virt_sid(p, virt_sid(current));
```

```
+ if (clone_flags & CLONE_THREAD) {
```

```
    p->tgid = current->tgid;
```

```
+ set_virt_tgid(p, virt_tgid(current));
```

```
+ }
```

```
if ((retval = security_task_alloc(p))
```

```
    goto bad_fork_cleanup_policy;
```

```

@@ -1181,6 +1188,9 @@ bad_fork_cleanup_cpuset:
    #endif
    cpuset_exit(p);
    bad_fork_cleanup:
+ if (virt_pid(p) != p->pid && virt_pid(p) > 0)
+ free_vpid(virt_pid(p));
+bad_fork_cleanup_module:
    if (p->binfmt)
        module_put(p->binfmt->module);
    bad_fork_cleanup_put_domain:
@@ -1206,7 +1216,7 @@ task_t * __devinit fork_idle(int cpu)
    task_t *task;
    struct pt_regs regs;

- task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL, 0);
+ task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL, 0, 0);
    if (!task)
        return ERR_PTR(-ENOMEM);
    init_idle(task, cpu);
@@ -1236,12 +1246,12 @@ static inline int fork_traceflag (unsign
    * It copies the process, and if successful kick-starts
    * it and waits for it to finish using the VM if required.
    */
-long do_fork(unsigned long clone_flags,
+static long do_fork_pid(unsigned long clone_flags,
                        unsigned long stack_start,
                        struct pt_regs *regs,
                        unsigned long stack_size,
                        int __user *parent_tidptr,
-                        int __user *child_tidptr)
+                        int __user *child_tidptr, int vpid)
    {
        struct task_struct *p;
        int trace = 0;
@@ -1255,7 +1265,8 @@ long do_fork(unsigned long clone_flags,
        clone_flags |= CLONE_PTRACE;
    }

- p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
+ p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr,
+ child_tidptr, pid, vpid);
    /*
     * Do this prior waking up the new thread - the thread pointer
     * might get invalid after that point, if the thread exits quickly.
@@ -1298,6 +1309,17 @@ long do_fork(unsigned long clone_flags,
    return pid;
}

```

```
+long do_fork(unsigned long clone_flags,  
+ unsigned long stack_start,  
+ struct pt_regs *regs,  
+ unsigned long stack_size,  
+ int __user *parent_tidptr,  
+ int __user *child_tidptr)  
+{  
+ return do_fork_pid(clone_flags, stack_start, regs, stack_size,  
+ parent_tidptr, child_tidptr, 0);  
+}  
+  
#ifndef ARCH_MIN_MMSTRUCT_ALIGN  
#define ARCH_MIN_MMSTRUCT_ALIGN 0  
#endif
```
