Subject: [RFC][PATCH 7/7] VPIDs: required VPS interface for VPIDs
Posted by Kirill Korotaev on Thu, 02 Feb 2006 16:32:56 GMT
View Forum Message <> Reply to Message

This patch adds small VPS/containers interface which is used by VPIDs

Kirill

```
--- ./include/linux/sched.h.vps_info 2006-02-02 14:58:53.000000000 +0300
+++ ./include/linux/sched.h 2006-02-02 18:38:15.134903248 +0300
@@ -15,6 +15,7 @@
 #include <linux/cpumask.h>
 #include <linux/errno.h>
 #include <linux/nodemask.h>
+#include <linux/vps_info.h>

 #include <asm/system.h>
 #include <asm/semaphore.h>
--- ./include/linux/vps_info.h.vps_info 2006-02-02 18:12:49.500834880 +0300
+++ ./include/linux/vps_info.h 2006-02-02 18:55:41.137886624 +0300
@@ -0,0 +1,44 @@
+#ifndef __VPS_INFO_H_
+#define __VPS_INFO_H_
+
+#include <linux/config.h>
+
+struct vps_common_info {
+ int id;
+ struct task_struct *init_task;
+ int is_sparse;
+ atomic_t counter;
+ atomic_t pcounter;
+};
+
+typedef struct vps_common_info *vps_info_t;
+
+#ifndef CONFIG_VPS
+extern struct vps_common_info super_vps_info;
+
+#define super_vps()  (&super_vps_info)
+#define current_vps()  (super_vps())
+#define task_vps(t)  (super_vps())
+#define inside_vps()  (0)
+#define task_inside_vps(t) (0)
+#define vps_stop(vps)  do { } while (0)
+#define vps_free(vps)  do { } while (0)
+#endif
+
```

```
+#define sparse_vpid(vps) ((vps)->is_sparse)
+#define set_sparse_vpid(vps) do { (vps)->is_sparse = 1; } while (0)
+
+#define vps_get(vps)  atomic_inc(&(vps)->counter)
+#define vps_put(vps)  do {     \
+  if (atomic_dec_and_test(&(vps)->counter)) \
+   vps_free(vps);    \
+ } while (0)
+
+#define vps_task_add(vps) atomic_inc(&(vps)->pcounter)
+#define vps_task_del(vps) do {     \
+  if (atomic_dec_and_test(&(vps)->pcounter)) \
+   vps_stop(vps);   \
+ } while (0)
+#define get_vps_tasks_num(vps) atomic_read(&(vps)->pcounter)
+
+#endif
--- ./init/main.c.vps_info 2006-02-02 14:15:35.000000000 +0300
+++ ./init/main.c 2006-02-02 18:40:36.592398440 +0300
@@ -47,6 +47,7 @@
 #include <linux/rmap.h>
 #include <linux/mempolicy.h>
 #include <linux/key.h>
+#include <linux/vps_info.h>

 #include <asm/io.h>
 #include <asm/bugs.h>
@@ -434,6 +435,14 @@ void __init parse_early_param(void)
  done = 1;
 }

+struct vps_common_info super_vps_info = {
+ .id  = 0,
+ .init_task = &init_task,
+ .counter = ATOMIC_INIT(1),
+ .pcounter = ATOMIC_INIT(1), /* init_task */
+ .is_sparse = 0,
+};
+
 /*
  * Activate the first processor.
  */
--- ./kernel/exit.c.vps_info 2006-02-02 14:33:58.000000000 +0300
+++ ./kernel/exit.c 2006-02-02 18:33:10.953145904 +0300
@@ -31,6 +31,7 @@
 #include <linux/signal.h>
 #include <linux/cn_proc.h>
 #include <linux/mutex.h>
```

```
+#include <linux/vps_info.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -107,6 +108,7 @@ repeat:
  spin_unlock(&p->proc_lock);
  proc_pid_flush(proc_dentry);
  release_thread(p);
+ vps_task_del(task_vps(p));
  put_task_struct(p);

  p = leader;
--- ./kernel/fork.c.vps_info 2006-02-02 14:41:40.000000000 +0300
+++ ./kernel/fork.c 2006-02-02 18:32:07.910729808 +0300
@@ -44,6 +44,7 @@
 #include <linux/rmap.h>
 #include <linux/acct.h>
 #include <linux/cn_proc.h>
+#include <linux/vps_info.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1139,6 +1140,7 @@ static task_t *copy_process(unsigned lon
  p->ioprio = current->ioprio;

  SET_LINKS(p);
+ vps_task_add(task_vps(p));
  if (unlikely(p->ptrace & PT_PTRACED))
   __ptrace_link(p, current->parent);

--- ./kernel/pid.c.vps_info 2006-02-02 14:58:34.000000000 +0300
+++ ./kernel/pid.c 2006-02-02 18:54:25.506384360 +0300
@@ -26,6 +26,7 @@
 #include <linux/init.h>
 #include <linux/bootmem.h>
 #include <linux/hash.h>
+#include <linux/vps_info.h>

 #ifdef CONFIG_VIRTUAL_PIDS
 static void __free_vpid(int vpid, struct task_struct *ve_tsk);
```