Subject: [RFC][PATCH 2/7] VPIDs: pid/vpid conversions
Posted by dev on Thu, 02 Feb 2006 16:21:13 GMT
View Forum Message <> Reply to Message

This is one of the major patches,
it adds vpid-to-pid conversions by placing macros
introduced in diff-vpid-macro patch.

Note that in CONFIG_VIRTUAL_PIDS=n case these macros expand to default code.

Kirill

```
--- ./drivers/char/tty_io.c.vpid_core 2006-02-02 14:15:33.271070768 +0300
+++ ./drivers/char/tty_io.c 2006-02-02 14:33:58.795005584 +0300
@@ -2366,7 +2366,7 @@ static int tiocgpgrp(struct tty_struct *
   */
  if (tty == real_tty && current->signal->tty != real_tty)
   return -ENOTTY;
- return put_user(real_tty->pgrp, p);
+ return put_user(pid_type_to_vpid(PIDTYPE_PGID, real_tty->pgrp), p);
 }

 static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t __user *p)
@@ -2386,6 +2386,9 @@ static int tiocspgrp(struct tty_struct *
   return -EFAULT;
  if (pgrp < 0)
   return -EINVAL;
+ pgrp = vpid_to_pid(pgrp);
+ if (pgrp < 0)
+  return -EPERM;
  if (session_of_pgrp(pgrp) != current->signal->session)
   return -EPERM;
  real_tty->pgrp = pgrp;
@@ -2402,7 +2405,7 @@ static int tiocgsid(struct tty_struct *t
   return -ENOTTY;
  if (real_tty->session <= 0)
   return -ENOTTY;
- return put_user(real_tty->session, p);
+ return put_user(pid_type_to_vpid(PIDTYPE_SID, real_tty->session), p);
 }

 static int tiocsetd(struct tty_struct *tty, int __user *p)
--- ./fs/binfmt_elf.c.vpid_core 2006-02-02 14:15:34.478887152 +0300
+++ ./fs/binfmt_elf.c 2006-02-02 14:33:58.797005280 +0300
@@ -1276,10 +1276,10 @@ static void fill_prstatus(struct elf_prs
  prstatus->pr_info.si_signo = prstatus->pr_cursig = signr;
  prstatus->pr_sigpend = p->pending.signal.sig[0];
  prstatus->pr_sighold = p->blocked.sig[0];
```

```
- prstatus->pr_pid = p->pid;
- prstatus->pr_ppid = p->parent->pid;
- prstatus->pr_pgrp = process_group(p);
- prstatus->pr_sid = p->signal->session;
+ prstatus->pr_pid = virt_pid(p);
+ prstatus->pr_ppid = virt_pid(p->parent);
+ prstatus->pr_pgrp = virt_pgid(p);
+ prstatus->pr_sid = virt_sid(p);
  if (thread_group_leader(p)) {
   /*
    * This is the record for the group leader.  Add in the
@@ -1322,10 +1322,10 @@ static int fill_psinfo(struct elf_prpsin
    psinfo->pr_psargs[i] = ' ';
  psinfo->pr_psargs[len] = 0;

- psinfo->pr_pid = p->pid;
- psinfo->pr_ppid = p->parent->pid;
- psinfo->pr_pgrp = process_group(p);
- psinfo->pr_sid = p->signal->session;
+ psinfo->pr_pid = virt_pid(p);
+ psinfo->pr_ppid = virt_pid(p->parent);
+ psinfo->pr_pgrp = virt_pgid(p);
+ psinfo->pr_sid = virt_sid(p);

  i = p->state ? ffz(~p->state) + 1 : 0;
  psinfo->pr_state = i;
--- ./fs/exec.c.vpid_core 2006-02-02 14:15:55.352713848 +0300
+++ ./fs/exec.c 2006-02-02 14:33:58.799004976 +0300
@@ -1282,7 +1282,7 @@ static void format_corename(char *corena
    case 'p':
     pid_in_pattern = 1;
     rc = snprintf(out_ptr, out_end - out_ptr,
-           "%d", current->tgid);
+           "%d", virt_tgid(current));
     if (rc > out_end - out_ptr)
      goto out;
     out_ptr += rc;
@@ -1354,7 +1354,7 @@ static void format_corename(char *corena
  if (!pid_in_pattern
            && (core_uses_pid || atomic_read(&current->mm->mm_users) != 1)) {
   rc = snprintf(out_ptr, out_end - out_ptr,
-         ".%d", current->tgid);
+         ".%d", virt_tgid(current));
   if (rc > out_end - out_ptr)
    goto out;
   out_ptr += rc;
--- ./fs/fcntl.c.vpid_core 2006-02-02 14:15:34.521880616 +0300
+++ ./fs/fcntl.c 2006-02-02 14:33:58.800004824 +0300
```

```
@@ -251,6 +251,7 @@ static int setfl(int fd, struct file * f
 static void f_modown(struct file *filp, unsigned long pid,
                      uid_t uid, uid_t euid, int force)
 {
+ pid = comb_vpid_to_pid(pid);
  write_lock_irq(&filp->f_owner.lock);
  if (force || !filp->f_owner.pid) {
   filp->f_owner.pid = pid;
@@ -317,7 +318,7 @@ static long do_fcntl(int fd, unsigned in
    * current syscall conventions, the only way
    * to fix this will be in libc.
    */
-  err = filp->f_owner.pid;
+  err = comb_pid_to_vpid(filp->f_owner.pid);
   force_successful_syscall_return();
   break;
  case F_SETOWN:
--- ./fs/locks.c.vpid_core 2006-02-02 14:15:34.551876056 +0300
+++ ./fs/locks.c 2006-02-02 14:33:58.801004672 +0300
@@ -1573,7 +1573,7 @@ int fcntl_getlk(struct file *filp, struc

  flock.l_type = F_UNLCK;
  if (fl != NULL) {
-  flock.l_pid = fl->fl_pid;
+  flock.l_pid = pid_type_to_vpid(PIDTYPE_TGID, fl->fl_pid);
 #if BITS_PER_LONG == 32
   /*
    * Make sure we can represent the posix lock via
@@ -1727,7 +1727,7 @@ int fcntl_getlk64(struct file *filp, str

  flock.l_type = F_UNLCK;
  if (fl != NULL) {
-  flock.l_pid = fl->fl_pid;
+  flock.l_pid = pid_type_to_vpid(PIDTYPE_TGID, fl->fl_pid);
   flock.l_start = fl->fl_start;
   flock.l_len = fl->fl_end == OFFSET_MAX ? 0 :
    fl->fl_end - fl->fl_start + 1;
--- ./fs/proc/array.c.vpid_core 2006-02-02 14:15:34.661859336 +0300
+++ ./fs/proc/array.c 2006-02-02 14:33:58.802004520 +0300
@@ -174,9 +174,10 @@ static inline char * task_state(struct t
   "Gid:\t%d\t%d\t%d\t%d\n",
   get_task_state(p),
   (p->sleep_avg/1024)*100/(1020000000/1024),
-        p->tgid,
-  p->pid, pid_alive(p) ? p->group_leader->real_parent->tgid : 0,
-  pid_alive(p) && p->ptrace ? p->parent->pid : 0,
+        get_task_tgid(p),
+  get_task_pid(p),
```

```
+  get_task_ppid(p),
+  pid_alive(p) && p->ptrace ? get_task_pid(p->parent) : 0,
   p->uid, p->euid, p->suid, p->fsuid,
   p->gid, p->egid, p->sgid, p->fsgid);
  read_unlock(&tasklist_lock);
@@ -370,11 +371,12 @@ static int do_task_stat(struct task_stru
  }
  if (task->signal) {
   if (task->signal->tty) {
-   tty_pgrp = task->signal->tty->pgrp;
+   tty_pgrp = pid_type_to_vpid(PIDTYPE_PGID,
+     task->signal->tty->pgrp);
    tty_nr = new_encode_dev(tty_devnum(task->signal->tty));
   }
-  pgid = process_group(task);
-  sid = task->signal->session;
+  pgid = get_task_pgid(task);
+  sid = get_task_sid(task);
   cmin_flt = task->signal->cmin_flt;
   cmaj_flt = task->signal->cmaj_flt;
   cutime = task->signal->cutime;
@@ -388,7 +390,7 @@ static int do_task_stat(struct task_stru
   }
   it_real_value = task->signal->real_timer.expires;
  }
- ppid = pid_alive(task) ? task->group_leader->real_parent->tgid : 0;
+ ppid = get_task_ppid(task);
  read_unlock(&tasklist_lock);

  if (!whole || num_threads<2)
@@ -415,7 +417,7 @@ static int do_task_stat(struct task_stru
  res = sprintf(buffer,"%d (%s) %c %d %d %d %d %d %lu %lu \
 %lu %lu %lu %lu %lu %ld %ld %ld %ld %d %ld %llu %lu %ld %lu %lu %lu %lu %lu \
 %lu %lu %lu %lu %lu %lu %lu %lu %d %d %lu %lu\n",
-  task->pid,
+  get_task_pid(task),
   tcomm,
   state,
   ppid,
--- ./fs/proc/base.c.vpid_core 2006-02-02 14:15:34.662859184 +0300
+++ ./fs/proc/base.c 2006-02-02 14:33:58.804004216 +0300
@@ -1879,14 +1879,14 @@ static int proc_self_readlink(struct den
         int buflen)
 {
  char tmp[30];
- sprintf(tmp, "%d", current->tgid);
+ sprintf(tmp, "%d", get_task_tgid(current));
  return vfs_readlink(dentry,buffer,buflen,tmp);
```

```
 }

 static void *proc_self_follow_link(struct dentry *dentry, struct nameidata *nd)
 {
  char tmp[30];
- sprintf(tmp, "%d", current->tgid);
+ sprintf(tmp, "%d", get_task_tgid(current));
  return ERR_PTR(vfs_follow_link(nd,tmp));
 }

@@ -2134,7 +2134,7 @@ static int get_tid_list(int index, unsig
   * via next_thread().
   */
  if (pid_alive(task)) do {
-  int tid = task->pid;
+  int tid = get_task_pid(task);

   if (--index >= 0)
    continue;
--- ./include/net/scm.h.vpid_core 2006-01-03 06:21:10.000000000 +0300
+++ ./include/net/scm.h 2006-02-02 14:33:58.805004064 +0300
@@ -40,7 +40,7 @@ static __inline__ int scm_send(struct so
  memset(scm, 0, sizeof(*scm));
  scm->creds.uid = current->uid;
  scm->creds.gid = current->gid;
- scm->creds.pid = current->tgid;
+ scm->creds.pid = virt_tgid(current);
  if (msg->msg_controllen <= 0)
   return 0;
  return __scm_send(sock, msg, scm);
--- ./ipc/msg.c.vpid_core 2006-02-02 14:15:35.148785312 +0300
+++ ./ipc/msg.c 2006-02-02 14:33:58.806003912 +0300
@@ -540,7 +540,7 @@ static inline int pipelined_send(struct
     msr->r_msg = ERR_PTR(-E2BIG);
    } else {
     msr->r_msg = NULL;
-    msq->q_lrpid = msr->r_tsk->pid;
+    msq->q_lrpid = virt_pid(msr->r_tsk);
     msq->q_rtime = get_seconds();
     wake_up_process(msr->r_tsk);
     smp_mb();
@@ -622,7 +622,7 @@ asmlinkage long sys_msgsnd (int msqid, s
   }
  }

- msq->q_lspid = current->tgid;
+ msq->q_lspid = virt_tgid(current);
  msq->q_stime = get_seconds();
```

```
  if(!pipelined_send(msq,msg)) {
@@ -718,7 +718,7 @@ asmlinkage long sys_msgrcv (int msqid, s
    list_del(&msg->m_list);
    msq->q_qnum--;
    msq->q_rtime = get_seconds();
-   msq->q_lrpid = current->tgid;
+   msq->q_lrpid = virt_tgid(current);
    msq->q_cbytes -= msg->m_ts;
    atomic_sub(msg->m_ts,&msg_bytes);
    atomic_dec(&msg_hdrs);
--- ./ipc/sem.c.vpid_core 2006-02-02 14:15:35.149785160 +0300
+++ ./ipc/sem.c 2006-02-02 14:33:58.807003760 +0300
@@ -743,7 +743,7 @@ static int semctl_main(int semid, int se
   for (un = sma->undo; un; un = un->id_next)
    un->semadj[semnum] = 0;
   curr->semval = val;
-  curr->sempid = current->tgid;
+  curr->sempid = virt_tgid(current);
   sma->sem_ctime = get_seconds();
   /* maybe some queued-up processes were waiting for this */
   update_queue(sma);
@@ -1135,7 +1135,7 @@ retry_undos:
  if (error)
   goto out_unlock_free;

- error = try_atomic_semop (sma, sops, nsops, un, current->tgid);
+ error = try_atomic_semop (sma, sops, nsops, un, virt_tgid(current));
  if (error <= 0) {
   if (alter && error == 0)
    update_queue (sma);
@@ -1150,7 +1150,7 @@ retry_undos:
  queue.sops = sops;
  queue.nsops = nsops;
  queue.undo = un;
- queue.pid = current->tgid;
+ queue.pid = virt_tgid(current);
  queue.id = semid;
  queue.alter = alter;
  if (alter)
@@ -1320,7 +1320,7 @@ found:
      sem->semval = 0;
     if (sem->semval > SEMVMX)
      sem->semval = SEMVMX;
-    sem->sempid = current->tgid;
+    sem->sempid = virt_tgid(current);
    }
   }
```

```
   sma->sem_otime = get_seconds();
--- ./ipc/shm.c.vpid_core 2006-02-02 14:15:35.149785160 +0300
+++ ./ipc/shm.c 2006-02-02 14:33:58.808003608 +0300
@@ -93,7 +93,7 @@ static inline void shm_inc (int id) {
  if(!(shp = shm_lock(id)))
   BUG();
  shp->shm_atim = get_seconds();
- shp->shm_lprid = current->tgid;
+ shp->shm_lprid = virt_tgid(current);
  shp->shm_nattch++;
  shm_unlock(shp);
 }
@@ -143,7 +143,7 @@ static void shm_close (struct vm_area_st
  /* remove from the list of attaches of the shm segment */
  if(!(shp = shm_lock(id)))
   BUG();
- shp->shm_lprid = current->tgid;
+ shp->shm_lprid = virt_tgid(current);
  shp->shm_dtim = get_seconds();
  shp->shm_nattch--;
  if(shp->shm_nattch == 0 &&
@@ -239,7 +239,7 @@ static int newseg (key_t key, int shmflg
  if(id == -1)
   goto no_id;

- shp->shm_cprid = current->tgid;
+ shp->shm_cprid = virt_tgid(current);
  shp->shm_lprid = 0;
  shp->shm_atim = shp->shm_dtim = 0;
  shp->shm_ctim = get_seconds();
--- ./kernel/capability.c.vpid_core 2006-02-02 14:15:35.152784704 +0300
+++ ./kernel/capability.c 2006-02-02 14:33:58.808003608 +0300
@@ -67,7 +67,7 @@ asmlinkage long sys_capget(cap_user_head
      spin_lock(&task_capability_lock);
      read_lock(&tasklist_lock);

-     if (pid && pid != current->pid) {
+     if (pid && pid != virt_pid(current)) {
       target = find_task_by_pid(pid);
       if (!target) {
           ret = -ESRCH;
@@ -100,6 +100,10 @@ static inline int cap_set_pg(int pgrp, k
  int ret = -EPERM;
  int found = 0;

+ pgrp = vpid_to_pid(pgrp);
+ if (pgrp < 0)
+  return ret;
```

```
+
  do_each_task_pid(pgrp, PIDTYPE_PGID, g) {
   target = g;
   while_each_thread(g, target) {
@@ -199,7 +203,7 @@ asmlinkage long sys_capset(cap_user_head
      spin_lock(&task_capability_lock);
      read_lock(&tasklist_lock);

-     if (pid > 0 && pid != current->pid) {
+     if (pid > 0 && pid != virt_pid(current)) {
          target = find_task_by_pid(pid);
          if (!target) {
              ret = -ESRCH;
--- ./kernel/exit.c.vpid_core 2006-02-02 14:15:35.156784096 +0300
+++ ./kernel/exit.c 2006-02-02 14:33:58.810003304 +0300
@@ -139,6 +139,8 @@ int session_of_pgrp(int pgrp)
  struct task_struct *p;
  int sid = -1;

+ WARN_ON(is_virtual_pid(pgrp));
+
  read_lock(&tasklist_lock);
  do_each_task_pid(pgrp, PIDTYPE_PGID, p) {
   if (p->signal->session > 0) {
@@ -168,10 +170,12 @@ static int will_become_orphaned_pgrp(int
  struct task_struct *p;
  int ret = 1;

+ WARN_ON(is_virtual_pid(pgrp));
+
  do_each_task_pid(pgrp, PIDTYPE_PGID, p) {
   if (p == ignored_task
     || p->exit_state
-    || p->real_parent->pid == 1)
+    || virt_pid(p->real_parent) == 1)
    continue;
   if (process_group(p->real_parent) != pgrp
       && p->real_parent->signal->session == p->signal->session) {
@@ -186,6 +190,8 @@ int is_orphaned_pgrp(int pgrp)
 {
  int retval;

+ WARN_ON(is_virtual_pid(pgrp));
+
  read_lock(&tasklist_lock);
  retval = will_become_orphaned_pgrp(pgrp, NULL);
  read_unlock(&tasklist_lock);
@@ -198,6 +204,8 @@ static int has_stopped_jobs(int pgrp)
```

```
  int retval = 0;
  struct task_struct *p;

+ WARN_ON(is_virtual_pid(pgrp));
+
  do_each_task_pid(pgrp, PIDTYPE_PGID, p) {
   if (p->state != TASK_STOPPED)
    continue;
@@ -263,6 +271,9 @@ void __set_special_pids(pid_t session, p
 {
  struct task_struct *curr = current->group_leader;

+ WARN_ON(is_virtual_pid(pgrp));
+ WARN_ON(is_virtual_pid(session));
+
  if (curr->signal->session != session) {
   detach_pid(curr, PIDTYPE_SID);
   curr->signal->session = session;
@@ -957,14 +968,19 @@ asmlinkage void sys_exit_group(int error
 static int eligible_child(pid_t pid, int options, task_t *p)
 {
  if (pid > 0) {
-  if (p->pid != pid)
+  if ((is_virtual_pid(pid) ? virt_pid(p) : p->pid) != pid)
    return 0;
  } else if (!pid) {
   if (process_group(p) != process_group(current))
    return 0;
  } else if (pid != -1) {
-  if (process_group(p) != -pid)
-   return 0;
+  if (__is_virtual_pid(-pid)) {
+   if (virt_pgid(p) != -pid)
+    return 0;
+  } else {
+   if (process_group(p) != -pid)
+    return 0;
+  }
  }

  /*
@@ -1154,7 +1170,7 @@ static int wait_task_zombie(task_t *p, i
   p->exit_state = EXIT_ZOMBIE;
   return retval;
  }
- retval = p->pid;
+ retval = get_task_pid(p);
  if (p->real_parent != p->parent) {
```

```
   write_lock_irq(&tasklist_lock);
   /* Double-check with lock held.  */
@@ -1289,7 +1305,7 @@ bail_ref:
  if (!retval && infop)
   retval = put_user(p->uid, &infop->si_uid);
  if (!retval)
-  retval = p->pid;
+  retval = get_task_pid(p);
  put_task_struct(p);

  BUG_ON(!retval);
--- ./kernel/fork.c.vpid_core 2006-02-02 14:15:55.472695608 +0300
+++ ./kernel/fork.c 2006-02-02 14:41:40.806769120 +0300
@@ -854,7 +854,7 @@ asmlinkage long sys_set_tid_address(int
 {
  current->clear_child_tid = tidptr;

- return current->pid;
+ return virt_pid(current);
 }

 /*
--- ./kernel/sched.c.vpid_core 2006-02-02 14:15:55.479694544 +0300
+++ ./kernel/sched.c 2006-02-02 14:33:58.815002544 +0300
@@ -1674,7 +1674,7 @@ asmlinkage void schedule_tail(task_t *pr
  preempt_enable();
 #endif
  if (current->set_child_tid)
-  put_user(current->pid, current->set_child_tid);
+  put_user(virt_pid(current), current->set_child_tid);
 }

 /*
--- ./kernel/signal.c.vpid_core 2006-02-02 14:15:55.481694240 +0300
+++ ./kernel/signal.c 2006-02-02 14:33:58.817002240 +0300
@@ -838,7 +838,7 @@ static int send_signal(int sig, struct s
    q->info.si_signo = sig;
    q->info.si_errno = 0;
    q->info.si_code = SI_USER;
-   q->info.si_pid = current->pid;
+   q->info.si_pid = virt_pid(current);
    q->info.si_uid = current->uid;
    break;
   case (unsigned long) SEND_SIG_PRIV:
@@ -1159,6 +1159,11 @@ int __kill_pg_info(int sig, struct sigin
  if (pgrp <= 0)
   return -EINVAL;
```

```
+ /* Use __vpid_to_pid(). This function is used under write_lock
+  * tasklist_lock. */
+ if (is_virtual_pid(pgrp))
+  pgrp = __vpid_to_pid(pgrp);
+
  success = 0;
  retval = -ESRCH;
  do_each_task_pid(pgrp, PIDTYPE_PGID, p) {
@@ -1254,7 +1259,7 @@ static int kill_something_info(int sig,

   read_lock(&tasklist_lock);
   for_each_process(p) {
-   if (p->pid > 1 && p->tgid != current->tgid) {
+   if (virt_pid(p) > 1 && p->tgid != current->tgid) {
     int err = group_send_sig_info(sig, info, p);
     ++count;
     if (err != -EPERM)
@@ -1564,7 +1569,7 @@ void do_notify_parent(struct task_struct

  info.si_signo = sig;
  info.si_errno = 0;
- info.si_pid = tsk->pid;
+ info.si_pid = get_task_pid_ve(tsk, tsk->parent);
  info.si_uid = tsk->uid;

  /* FIXME: find out whether or not this is supposed to be c*time. */
@@ -1629,7 +1634,7 @@ static void do_notify_parent_cldstop(str

  info.si_signo = SIGCHLD;
  info.si_errno = 0;
- info.si_pid = tsk->pid;
+ info.si_pid = get_task_pid_ve(tsk, parent);
  info.si_uid = tsk->uid;

  /* FIXME: find out whether or not this is supposed to be c*time. */
@@ -1732,7 +1737,7 @@ void ptrace_notify(int exit_code)
  memset(&info, 0, sizeof info);
  info.si_signo = SIGTRAP;
  info.si_code = exit_code;
- info.si_pid = current->pid;
+ info.si_pid = virt_pid(current);
  info.si_uid = current->uid;

  /* Let the debugger run.  */
@@ -1957,7 +1962,7 @@ relock:
     info->si_signo = signr;
     info->si_errno = 0;
     info->si_code = SI_USER;
```

```
-    info->si_pid = current->parent->pid;
+    info->si_pid = virt_pid(current->parent);
     info->si_uid = current->parent->uid;
    }

@@ -2340,7 +2345,7 @@ sys_kill(int pid, int sig)
  info.si_signo = sig;
  info.si_errno = 0;
  info.si_code = SI_USER;
- info.si_pid = current->tgid;
+ info.si_pid = virt_tgid(current);
  info.si_uid = current->uid;

  return kill_something_info(sig, &info, pid);
@@ -2356,12 +2361,12 @@ static int do_tkill(int tgid, int pid, i
  info.si_signo = sig;
  info.si_errno = 0;
  info.si_code = SI_TKILL;
- info.si_pid = current->tgid;
+ info.si_pid = virt_tgid(current);
  info.si_uid = current->uid;

  read_lock(&tasklist_lock);
  p = find_task_by_pid(pid);
- if (p && (tgid <= 0 || p->tgid == tgid)) {
+ if (p && (tgid <= 0 || virt_tgid(p) == tgid)) {
   error = check_kill_permission(sig, &info, p);
   /*
    * The null signal is a permissions and process existence
--- ./kernel/sys.c.vpid_core 2006-02-02 14:15:55.482694088 +0300
+++ ./kernel/sys.c 2006-02-02 14:39:46.597131624 +0300
@@ -281,7 +281,7 @@ asmlinkage long sys_setpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (!who)
-    who = current->pid;
+    who = virt_pid(current);
    p = find_task_by_pid(who);
    if (p)
     error = set_one_prio(p, niceval, error);
@@ -289,6 +289,8 @@ asmlinkage long sys_setpriority(int whic
   case PRIO_PGRP:
    if (!who)
     who = process_group(current);
+   else
+    who = vpid_to_pid(who);
    do_each_task_pid(who, PIDTYPE_PGID, p) {
     error = set_one_prio(p, niceval, error);
```

```
    } while_each_task_pid(who, PIDTYPE_PGID, p);
@@ -334,7 +336,7 @@ asmlinkage long sys_getpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (!who)
-    who = current->pid;
+    who = virt_pid(current);
    p = find_task_by_pid(who);
    if (p) {
     niceval = 20 - task_nice(p);
@@ -345,6 +347,8 @@ asmlinkage long sys_getpriority(int whic
   case PRIO_PGRP:
    if (!who)
     who = process_group(current);
+   else
+    who = vpid_to_pid(who);
    do_each_task_pid(who, PIDTYPE_PGID, p) {
     niceval = 20 - task_nice(p);
     if (niceval > retval)
@@ -1100,9 +1104,10 @@ asmlinkage long sys_setpgid(pid_t pid, p
  struct task_struct *p;
  struct task_struct *group_leader = current->group_leader;
  int err = -EINVAL;
+ pid_t _pgid;

  if (!pid)
-  pid = group_leader->pid;
+  pid = virt_pid(group_leader);
  if (!pgid)
   pgid = pid;
  if (pgid < 0)
@@ -1139,12 +1144,15 @@ asmlinkage long sys_setpgid(pid_t pid, p
  if (p->signal->leader)
   goto out;

+ _pgid = virt_pid(p);
  if (pgid != pid) {
   struct task_struct *p;

   do_each_task_pid(pgid, PIDTYPE_PGID, p) {
-   if (p->signal->session == group_leader->signal->session)
+   if (p->signal->session == group_leader->signal->session) {
+    _pgid = virt_pid(p);
     goto ok_pgid;
+   }
   } while_each_task_pid(pgid, PIDTYPE_PGID, p);
   goto out;
  }
```

```
@@ -1157,7 +1165,14 @@ ok_pgid:
  if (process_group(p) != pgid) {
   detach_pid(p, PIDTYPE_PGID);
   p->signal->pgrp = pgid;
+  set_virt_pgid(p, _pgid);
   attach_pid(p, PIDTYPE_PGID, pgid);
+  if (atomic_read(&p->signal->count) != 1) {
+   task_t *t;
+   for (t = next_thread(p); t != p; t = next_thread(t)) {
+    set_virt_pgid(t, _pgid);
+   }
+  }
  }

  err = 0;
@@ -1170,7 +1185,7 @@ out:
 asmlinkage long sys_getpgid(pid_t pid)
 {
  if (!pid) {
-  return process_group(current);
+  return virt_pgid(current);
  } else {
   int retval;
   struct task_struct *p;
@@ -1182,7 +1197,7 @@ asmlinkage long sys_getpgid(pid_t pid)
   if (p) {
    retval = security_task_getpgid(p);
    if (!retval)
-    retval = process_group(p);
+    retval = virt_pgid(p);
   }
   read_unlock(&tasklist_lock);
   return retval;
@@ -1194,7 +1209,7 @@ asmlinkage long sys_getpgid(pid_t pid)
 asmlinkage long sys_getpgrp(void)
 {
  /* SMP - assuming writes are word atomic this is fine */
- return process_group(current);
+ return virt_pgid(current);
 }

 #endif
@@ -1202,7 +1217,7 @@ asmlinkage long sys_getpgrp(void)
 asmlinkage long sys_getsid(pid_t pid)
 {
  if (!pid) {
-  return current->signal->session;
+  return virt_sid(current);
```

```
  } else {
   int retval;
   struct task_struct *p;
@@ -1214,7 +1229,7 @@ asmlinkage long sys_getsid(pid_t pid)
   if(p) {
    retval = security_task_getsid(p);
    if (!retval)
-    retval = p->signal->session;
+    retval = virt_sid(p);
   }
   read_unlock(&tasklist_lock);
   return retval;
@@ -1236,9 +1251,11 @@ asmlinkage long sys_setsid(void)

  group_leader->signal->leader = 1;
  __set_special_pids(group_leader->pid, group_leader->pid);
+ set_virt_pgid(group_leader, virt_pid(current));
+ set_virt_sid(group_leader, virt_pid(current));
  group_leader->signal->tty = NULL;
  group_leader->signal->tty_old_pgrp = 0;
- err = process_group(group_leader);
+ err = virt_pgid(group_leader);
 out:
  write_unlock_irq(&tasklist_lock);
  up(&tty_sem);
--- ./kernel/timer.c.vpid_core 2006-02-02 14:15:35.185779688 +0300
+++ ./kernel/timer.c 2006-02-02 14:33:58.821001632 +0300
@@ -943,7 +943,7 @@ asmlinkage unsigned long sys_alarm(unsig
  */
 asmlinkage long sys_getpid(void)
 {
- return current->tgid;
+ return virt_tgid(current);
 }

 /*
@@ -970,7 +970,7 @@ asmlinkage long sys_getppid(void)

  parent = me->group_leader->real_parent;
  for (;;) {
-  pid = parent->tgid;
+  pid = virt_tgid(parent);
 #if defined(CONFIG_SMP) || defined(CONFIG_PREEMPT)
 {
   struct task_struct *old = parent;
@@ -1117,7 +1117,7 @@ EXPORT_SYMBOL(schedule_timeout_uninterru
 /* Thread ID - the internal kernel "pid" */
 asmlinkage long sys_gettid(void)
```

```
 {
- return current->pid;
+ return virt_pid(current);
 }

 /*
--- ./net/core/scm.c.vpid_core 2006-02-02 14:15:35.252769504 +0300
+++ ./net/core/scm.c 2006-02-02 14:33:58.822001480 +0300
@@ -42,7 +42,7 @@

 static __inline__ int scm_check_creds(struct ucred *creds)
 {
- if ((creds->pid == current->tgid || capable(CAP_SYS_ADMIN)) &&
+ if ((creds->pid == virt_tgid(current) || capable(CAP_SYS_ADMIN)) &&
      ((creds->uid == current->uid || creds->uid == current->euid ||
        creds->uid == current->suid) || capable(CAP_SETUID)) &&
      ((creds->gid == current->gid || creds->gid == current->egid ||
--- ./net/unix/af_unix.c.vpid_core 2006-02-02 14:15:35.505731048 +0300
+++ ./net/unix/af_unix.c 2006-02-02 14:33:58.823001328 +0300
@@ -439,7 +439,7 @@ static int unix_listen(struct socket *so
  sk->sk_max_ack_backlog = backlog;
  sk->sk_state  = TCP_LISTEN;
  /* set credentials so connect can copy them */
- sk->sk_peercred.pid = current->tgid;
+ sk->sk_peercred.pid = virt_tgid(current);
  sk->sk_peercred.uid = current->euid;
  sk->sk_peercred.gid = current->egid;
  err = 0;
@@ -1043,7 +1043,7 @@ restart:
  unix_peer(newsk) = sk;
  newsk->sk_state  = TCP_ESTABLISHED;
  newsk->sk_type  = sk->sk_type;
- newsk->sk_peercred.pid = current->tgid;
+ newsk->sk_peercred.pid = virt_tgid(current);
  newsk->sk_peercred.uid = current->euid;
  newsk->sk_peercred.gid = current->egid;
  newu = unix_sk(newsk);
@@ -1107,7 +1107,7 @@ static int unix_socketpair(struct socket
  sock_hold(skb);
  unix_peer(ska)=skb;
  unix_peer(skb)=ska;
- ska->sk_peercred.pid = skb->sk_peercred.pid = current->tgid;
+ ska->sk_peercred.pid = skb->sk_peercred.pid = virt_tgid(current);
  ska->sk_peercred.uid = skb->sk_peercred.uid = current->euid;
  ska->sk_peercred.gid = skb->sk_peercred.gid = current->egid;
```