Subject: [PATCH] Use helpers to obtain task pid in printks (arch code)
Posted by Pavel Emelianov on Mon, 27 Aug 2007 07:30:06 GMT
View Forum Message <> Reply to Message

From: Alexey Dobriyan <adobriyan@openvz.org>

One of the easiest things to isolate is the pid printed in kernel
log. There was a patch, that made this for arch-independent code,
this one makes so for arch/xxx files.

It took some time to cross-compile it, but hopefully these are all
the printks in arch code.

Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>
Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
 arch/alpha/kernel/semaphore.c          |   16 +--
 arch/alpha/kernel/traps.c              |    6 -
 arch/alpha/mm/fault.c                  |    2
 arch/arm/kernel/process.c              |    2
 arch/arm/kernel/ptrace.c               |    8 -
 arch/arm/kernel/traps.c                |   10 -
 arch/arm/mm/alignment.c                |    2
 arch/i386/kernel/process.c             |    2
 arch/i386/kernel/signal.c              |    4
 arch/i386/kernel/traps.c               |    4
 arch/i386/mm/fault.c                   |    4
 arch/ia64/ia32/sys_ia32.c              |    6 -
 arch/ia64/kernel/perfmon.c             |  161 ++++++++++++++++----------------
 arch/ia64/kernel/perfmon_default_smpl.c |    8 -
 arch/ia64/kernel/process.c             |    3
 arch/ia64/kernel/traps.c               |    6 -
 arch/ia64/kernel/unaligned.c           |    5
 arch/m32r/kernel/traps.c               |    2
 arch/m68k/kernel/traps.c               |    4
 arch/mips/kernel/traps.c               |    2
 arch/parisc/kernel/traps.c             |   10 -
 arch/parisc/kernel/unaligned.c         |    2
 arch/parisc/mm/fault.c                 |    2
 arch/powerpc/kernel/process.c          |    2
 arch/powerpc/kernel/traps.c            |    2
 arch/s390/kernel/process.c             |    2
 arch/sh/kernel/process.c               |    2
 arch/sh/kernel/signal.c                |    4
 arch/sh/kernel/traps.c                 |    7 -
 arch/sh64/kernel/traps.c               |    4
```

```
 arch/sh64/mm/fault.c                  |    4
 arch/sparc/kernel/ptrace.c            |    4
 arch/sparc/kernel/sys_sparc.c         |    2
 arch/sparc/kernel/traps.c             |    4
 arch/sparc64/kernel/traps.c           |    2
 arch/um/sys-x86_64/sysrq.c            |    4
 arch/xtensa/kernel/traps.c            |    6 -
 37 files changed, 163 insertions(+), 157 deletions(-)

diff --git a/arch/alpha/kernel/semaphore.c b/arch/alpha/kernel/semaphore.c
--- a/arch/alpha/kernel/semaphore.c
+++ b/arch/alpha/kernel/semaphore.c
@@ -69,7 +69,7 @@ __down_failed(struct semaphore *sem)

 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down failed(%p)\n",
-        tsk->comm, tsk->pid, sem);
+        tsk->comm, task_pid_nr(tsk), sem);
 #endif

  tsk->state = TASK_UNINTERRUPTIBLE;
@@ -98,7 +98,7 @@ __down_failed(struct semaphore *sem)

 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down acquired(%p)\n",
-        tsk->comm, tsk->pid, sem);
+        tsk->comm, task_pid_nr(tsk), sem);
 #endif
 }

@@ -111,7 +111,7 @@ __down_failed_interruptible(struct semaphore *sem)

 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down failed(%p)\n",
-        tsk->comm, tsk->pid, sem);
+        tsk->comm, task_pid_nr(tsk), sem);
 #endif

  tsk->state = TASK_INTERRUPTIBLE;
@@ -139,7 +139,7 @@ __down_failed_interruptible(struct semaphore *sem)

 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down %s(%p)\n",
-        current->comm, current->pid,
+        current->comm, task_pid_nr(current),
         (ret < 0 ? "interrupted" : "acquired"), sem);
 #endif
  return ret;
```

```
@@ -168,7 +168,7 @@ down(struct semaphore *sem)
 #endif
 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down(%p) <count=%d> from %p\n",
-        current->comm, current->pid, sem,
+        current->comm, task_pid_nr(current), sem,
         atomic_read(&sem->count), __builtin_return_address(0));
 #endif
  __down(sem);
@@ -182,7 +182,7 @@ down_interruptible(struct semaphore *sem)
 #endif
 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down(%p) <count=%d> from %p\n",
-        current->comm, current->pid, sem,
+        current->comm, task_pid_nr(current), sem,
         atomic_read(&sem->count), __builtin_return_address(0));
 #endif
  return __down_interruptible(sem);
@@ -201,7 +201,7 @@ down_trylock(struct semaphore *sem)

 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): down_trylock %s from %p\n",
-        current->comm, current->pid,
+        current->comm, task_pid_nr(current),
         ret ? "failed" : "acquired",
         __builtin_return_address(0));
 #endif
@@ -217,7 +217,7 @@ up(struct semaphore *sem)
 #endif
 #ifdef CONFIG_DEBUG_SEMAPHORE
  printk("%s(%d): up(%p) <count=%d> from %p\n",
-        current->comm, current->pid, sem,
+        current->comm, task_pid_nr(current), sem,
         atomic_read(&sem->count), __builtin_return_address(0));
 #endif
  __up(sem);
diff --git a/arch/alpha/kernel/traps.c b/arch/alpha/kernel/traps.c
--- a/arch/alpha/kernel/traps.c
+++ b/arch/alpha/kernel/traps.c
@@ -182,7 +182,7 @@ die_if_kernel(char * str, struct pt_regs *regs, long err, unsigned long
*r9_15)
 #ifdef CONFIG_SMP
  printk("CPU %d ", hard_smp_processor_id());
 #endif
- printk("%s(%d): %s %ld\n", current->comm, current->pid, str, err);
+ printk("%s(%d): %s %ld\n", current->comm, task_pid_nr(current), str, err);
  dik_show_regs(regs, r9_15);
  add_taint(TAINT_DIE);
```

```
  dik_show_trace((unsigned long *)(regs+1));
@@ -646,7 +646,7 @@ got_exception:
  lock_kernel();

  printk("%s(%d): unhandled unaligned exception\n",
-        current->comm, current->pid);
+        current->comm, task_pid_nr(current));

  printk("pc = [<%016lx>]  ra = [<%016lx>]  ps = %04lx\n",
         pc, una_reg(26), regs->ps);
@@ -786,7 +786,7 @@ do_entUnaUser(void __user * va, unsigned long opcode,
   }
   if (++cnt < 5) {
    printk("%s(%d): unaligned trap at %016lx: %p %lx %ld\n",
-          current->comm, current->pid,
+          current->comm, task_pid_nr(current),
           regs->pc - 4, va, opcode, reg);
   }
   last_time = jiffies;
diff --git a/arch/alpha/mm/fault.c b/arch/alpha/mm/fault.c
--- a/arch/alpha/mm/fault.c
+++ b/arch/alpha/mm/fault.c
@@ -194,7 +194,7 @@ do_page_fault(unsigned long address, unsigned long mmcsr,
   goto survive;
  }
  printk(KERN_ALERT "VM: killing process %s(%d)\n",
-        current->comm, current->pid);
+        current->comm, task_pid_nr(current));
  if (!user_mode(regs))
   goto no_context;
  do_group_exit(SIGKILL);
diff --git a/arch/arm/kernel/process.c b/arch/arm/kernel/process.c
--- a/arch/arm/kernel/process.c
+++ b/arch/arm/kernel/process.c
@@ -265,7 +265,7 @@ void __show_regs(struct pt_regs *regs)
 void show_regs(struct pt_regs * regs)
 {
  printk("\n");
- printk("Pid: %d, comm: %20s\n", current->pid, current->comm);
+ printk("Pid: %d, comm: %20s\n", task_pid_nr(current), current->comm);
  __show_regs(regs);
  __backtrace();
 }
diff --git a/arch/arm/kernel/ptrace.c b/arch/arm/kernel/ptrace.c
--- a/arch/arm/kernel/ptrace.c
+++ b/arch/arm/kernel/ptrace.c
@@ -382,16 +382,16 @@ static void clear_breakpoint(struct task_struct *task, struct
debug_entry *bp)
```

```
   if (ret != 2 || old_insn.thumb != BREAKINST_THUMB)
    printk(KERN_ERR "%s:%d: corrupted Thumb breakpoint at "
-    "0x%08lx (0x%04x)\n", task->comm, task->pid,
-    addr, old_insn.thumb);
+    "0x%08lx (0x%04x)\n", task->comm,
+    task_pid_nr(task), addr, old_insn.thumb);
  } else {
   ret = swap_insn(task, addr & ~3, &old_insn.arm,
     &bp->insn.arm, 4);

   if (ret != 4 || old_insn.arm != BREAKINST_ARM)
    printk(KERN_ERR "%s:%d: corrupted ARM breakpoint at "
-    "0x%08lx (0x%08x)\n", task->comm, task->pid,
-    addr, old_insn.arm);
+    "0x%08lx (0x%08x)\n", task->comm,
+    task_pid_nr(task), addr, old_insn.arm);
  }
 }

diff --git a/arch/arm/kernel/traps.c b/arch/arm/kernel/traps.c
--- a/arch/arm/kernel/traps.c
+++ b/arch/arm/kernel/traps.c
@@ -223,7 +223,7 @@ static void __die(const char *str, int err, struct thread_info *thread, struct
p
  print_modules();
  __show_regs(regs);
  printk("Process %s (pid: %d, stack limit = 0x%p)\n",
-  tsk->comm, tsk->pid, thread + 1);
+  tsk->comm, task_pid_nr(tsk), thread + 1);

  if (!user_mode(regs) || in_interrupt()) {
   dump_mem("Stack: ", regs->ARM_sp,
@@ -341,7 +341,7 @@ asmlinkage void __exception do_undefinstr(struct pt_regs *regs)
 #ifdef CONFIG_DEBUG_USER
  if (user_debug & UDBG_UNDEFINED) {
   printk(KERN_INFO "%s (%d): undefined instruction: pc=%p\n",
-   current->comm, current->pid, pc);
+   current->comm, task_pid_nr(current), pc);
   dump_instr(regs);
  }
 #endif
@@ -392,7 +392,7 @@ static int bad_syscall(int n, struct pt_regs *regs)
 #ifdef CONFIG_DEBUG_USER
  if (user_debug & UDBG_SYSCALL) {
   printk(KERN_ERR "[%d] %s: obsolete system call %08x.\n",
-   current->pid, current->comm, n);
+   task_pid_nr(current), current->comm, n);
```

```
   dump_instr(regs);
  }
 #endif
@@ -569,7 +569,7 @@ asmlinkage int arm_syscall(int no, struct pt_regs *regs)
    */
   if (user_debug & UDBG_SYSCALL) {
    printk("[%d] %s: arm syscall %d\n",
-          current->pid, current->comm, no);
+          task_pid_nr(current), current->comm, no);
    dump_instr(regs);
    if (user_mode(regs)) {
     __show_regs(regs);
@@ -646,7 +646,7 @@ baddataabort(int code, unsigned long instr, struct pt_regs *regs)
 #ifdef CONFIG_DEBUG_USER
   if (user_debug & UDBG_BADABORT) {
    printk(KERN_ERR "[%d] %s: bad data abort: code %d instr 0x%08lx\n",
-    current->pid, current->comm, code, instr);
+    task_pid_nr(current), current->comm, code, instr);
    dump_instr(regs);
    show_pte(current->mm, addr);
   }
diff --git a/arch/arm/mm/alignment.c b/arch/arm/mm/alignment.c
--- a/arch/arm/mm/alignment.c
+++ b/arch/arm/mm/alignment.c
@@ -757,7 +757,7 @@ do_alignment(unsigned long addr, unsigned int fsr, struct pt_regs *regs)
   if (ai_usermode & 1)
    printk("Alignment trap: %s (%d) PC=0x%08lx Instr=0x%0*lx "
           "Address=0x%08lx FSR 0x%03x\n", current->comm,
-    current->pid, instrptr,
+    task_pid_nr(current), instrptr,
           thumb_mode(regs) ? 4 : 8,
           thumb_mode(regs) ? tinstr : instr,
           addr, fsr);
diff --git a/arch/i386/kernel/process.c b/arch/i386/kernel/process.c
--- a/arch/i386/kernel/process.c
+++ b/arch/i386/kernel/process.c
@@ -316,7 +316,7 @@ void __show_registers(struct pt_regs *regs, int all)

   printk("\n");
   printk("Pid: %d, comm: %.*s %s (%s %.*s)\n",
-    current->pid, TASK_COMM_LEN, current->comm,
+    task_pid_nr(current), TASK_COMM_LEN, current->comm,
     print_tainted(), init_utsname()->release,
     (int)strcspn(init_utsname()->version, " "),
     init_utsname()->version);
diff --git a/arch/i386/kernel/signal.c b/arch/i386/kernel/signal.c
--- a/arch/i386/kernel/signal.c
+++ b/arch/i386/kernel/signal.c
```

```
@@ -202,8 +202,8 @@ badframe:
  if (show_unhandled_signals && printk_ratelimit())
   printk("%s%s[%d] bad frame in sigreturn frame:%p eip:%lx"
          " esp:%lx oeax:%lx\n",
-      current->pid > 1 ? KERN_INFO : KERN_EMERG,
-      current->comm, current->pid, frame, regs->eip,
+      task_pid_nr(current) > 1 ? KERN_INFO : KERN_EMERG,
+      current->comm, task_pid_nr(current), frame, regs->eip,
       regs->esp, regs->orig_eax);

  force_sig(SIGSEGV, current);
diff --git a/arch/i386/kernel/traps.c b/arch/i386/kernel/traps.c
--- a/arch/i386/kernel/traps.c
+++ b/arch/i386/kernel/traps.c
@@ -354,7 +354,7 @@ void show_registers(struct pt_regs *regs)
  print_modules();
  __show_registers(regs, 0);
  printk(KERN_EMERG "Process %.*s (pid: %d, ti=%p task=%p task.ti=%p)",
-  TASK_COMM_LEN, current->comm, current->pid,
+  TASK_COMM_LEN, current->comm, task_pid_nr(current),
   current_thread_info(), current, task_thread_info(current));
  /*
   * When in-kernel, we also print out the stack and code at the
@@ -659,7 +659,7 @@ fastcall void __kprobes do_general_protection(struct pt_regs * regs,
      printk_ratelimit())
   printk(KERN_INFO
       "%s[%d] general protection eip:%lx esp:%lx error:%lx\n",
-      current->comm, current->pid,
+      current->comm, task_pid_nr(current),
       regs->eip, regs->esp, error_code);

  force_sig(SIGSEGV, current);
diff --git a/arch/i386/mm/fault.c b/arch/i386/mm/fault.c
--- a/arch/i386/mm/fault.c
+++ b/arch/i386/mm/fault.c
@@ -471,8 +471,8 @@ bad_area_nosemaphore:
       printk_ratelimit()) {
    printk("%s%s[%d]: segfault at %08lx eip %08lx "
       "esp %08lx error %lx\n",
-       tsk->pid > 1 ? KERN_INFO : KERN_EMERG,
-       tsk->comm, tsk->pid, address, regs->eip,
+       task_pid_nr(tsk) > 1 ? KERN_INFO : KERN_EMERG,
+       tsk->comm, task_pid_nr(tsk), address, regs->eip,
        regs->esp, error_code);
   }
   tsk->thread.cr2 = address;
diff --git a/arch/ia64/ia32/sys_ia32.c b/arch/ia64/ia32/sys_ia32.c
--- a/arch/ia64/ia32/sys_ia32.c
```

```
+++ b/arch/ia64/ia32/sys_ia32.c
@@ -773,7 +773,7 @@ emulate_mmap (struct file *file, unsigned long start, unsigned long len,
int pro
    if (flags & MAP_SHARED)
     printk(KERN_INFO
            "%s(%d): emulate_mmap() can't share head (addr=0x%lx)\n",
-           current->comm, current->pid, start);
+           current->comm, task_pid_nr(current), start);
    ret = mmap_subpage(file, start, min(PAGE_ALIGN(start), end), prot, flags,
         off);
    if (IS_ERR((void *) ret))
@@ -786,7 +786,7 @@ emulate_mmap (struct file *file, unsigned long start, unsigned long len,
int pro
    if (flags & MAP_SHARED)
     printk(KERN_INFO
            "%s(%d): emulate_mmap() can't share tail (end=0x%lx)\n",
-           current->comm, current->pid, end);
+           current->comm, task_pid_nr(current), end);
    ret = mmap_subpage(file, max(start, PAGE_START(end)), end, prot, flags,
         (off + len) - offset_in_page(end));
    if (IS_ERR((void *) ret))
@@ -816,7 +816,7 @@ emulate_mmap (struct file *file, unsigned long start, unsigned long len,
int pro

  if ((flags & MAP_SHARED) && !is_congruent)
   printk(KERN_INFO "%s(%d): emulate_mmap() can't share contents of incongruent mmap "
-         "(addr=0x%lx,off=0x%llx)\n", current->comm, current->pid, start, off);
+         "(addr=0x%lx,off=0x%llx)\n", current->comm, task_pid_nr(current), start, off);

  DBG("mmap_body: mapping [0x%lx-0x%lx) %s with poff 0x%llx\n", pstart, pend,
      is_congruent ? "congruent" : "not congruent", poff);
diff --git a/arch/ia64/kernel/perfmon.c b/arch/ia64/kernel/perfmon.c
--- a/arch/ia64/kernel/perfmon.c
+++ b/arch/ia64/kernel/perfmon.c
@@ -158,14 +158,14 @@
  */
 #define PROTECT_CTX(c, f) \
  do {  \
-  DPRINT(("spinlock_irq_save ctx %p by [%d]\n", c, current->pid)); \
+  DPRINT(("spinlock_irq_save ctx %p by [%d]\n", c, task_pid_nr(current))); \
   spin_lock_irqsave(&(c)->ctx_lock, f); \
-  DPRINT(("spinlocked ctx %p  by [%d]\n", c, current->pid)); \
+  DPRINT(("spinlocked ctx %p  by [%d]\n", c, task_pid_nr(current))); \
  } while(0)

 #define UNPROTECT_CTX(c, f) \
  do { \
-  DPRINT(("spinlock_irq_restore ctx %p by [%d]\n", c, current->pid)); \
```

```
+  DPRINT(("spinlock_irq_restore ctx %p by [%d]\n", c, task_pid_nr(current))); \
   spin_unlock_irqrestore(&(c)->ctx_lock, f); \
  } while(0)

@@ -227,12 +227,12 @@
 #ifdef PFM_DEBUGGING
 #define DPRINT(a) \
  do { \
-  if (unlikely(pfm_sysctl.debug >0)) { printk("%s.%d: CPU%d [%d] ", __FUNCTION__, __LINE__,
smp_processor_id(), current->pid); printk a; } \
+  if (unlikely(pfm_sysctl.debug >0)) { printk("%s.%d: CPU%d [%d] ", __FUNCTION__, __LINE__,
smp_processor_id(), task_pid_nr(current)); printk a; } \
  } while (0)

 #define DPRINT_ovfl(a) \
  do { \
-  if (unlikely(pfm_sysctl.debug > 0 && pfm_sysctl.debug_ovfl >0)) { printk("%s.%d: CPU%d [%d]
", __FUNCTION__, __LINE__, smp_processor_id(), current->pid); printk a; } \
+  if (unlikely(pfm_sysctl.debug > 0 && pfm_sysctl.debug_ovfl >0)) { printk("%s.%d: CPU%d [%d]
", __FUNCTION__, __LINE__, smp_processor_id(), task_pid_nr(current)); printk a; } \
  } while (0)
 #endif

@@ -913,7 +913,7 @@ pfm_mask_monitoring(struct task_struct *task)
  unsigned long mask, val, ovfl_mask;
  int i;

- DPRINT_ovfl(("masking monitoring for [%d]\n", task->pid));
+ DPRINT_ovfl(("masking monitoring for [%d]\n", task_pid_nr(task)));

  ovfl_mask = pmu_conf->ovfl_val;
  /*
@@ -992,12 +992,12 @@ pfm_restore_monitoring(struct task_struct *task)
  ovfl_mask = pmu_conf->ovfl_val;

  if (task != current) {
-  printk(KERN_ERR "perfmon.%d: invalid task[%d] current[%d]\n", __LINE__, task->pid,
current->pid);
+  printk(KERN_ERR "perfmon.%d: invalid task[%d] current[%d]\n", __LINE__, task_pid_nr(task),
task_pid_nr(current));
   return;
  }
  if (ctx->ctx_state != PFM_CTX_MASKED) {
   printk(KERN_ERR "perfmon.%d: task[%d] current[%d] invalid state=%d\n", __LINE__,
-   task->pid, current->pid, ctx->ctx_state);
+   task_pid_nr(task), task_pid_nr(current), ctx->ctx_state);
   return;
  }
```

```
  psr = pfm_get_psr();
@@ -1051,7 +1051,8 @@ pfm_restore_monitoring(struct task_struct *task)
    if ((mask & 0x1) == 0UL) continue;
    ctx->th_pmcs[i] = ctx->ctx_pmcs[i];
    ia64_set_pmc(i, ctx->th_pmcs[i]);
-   DPRINT(("[%d] pmc[%d]=0x%lx\n", task->pid, i, ctx->th_pmcs[i]));
+   DPRINT(("[%d] pmc[%d]=0x%lx\n",
+       task_pid_nr(task), i, ctx->th_pmcs[i]));
   }
   ia64_srlz_d();

@@ -1370,7 +1371,7 @@ pfm_reserve_session(struct task_struct *task, int is_syswide, unsigned
int cpu)

  error_conflict:
   DPRINT(("system wide not possible, conflicting session [%d] on CPU%d\n",
-      pfm_sessions.pfs_sys_session[cpu]->pid,
+      task_pid_nr(pfm_sessions.pfs_sys_session[cpu]),
     cpu));
  abort:
   UNLOCK_PFS(flags);
@@ -1442,7 +1443,7 @@ pfm_remove_smpl_mapping(struct task_struct *task, void *vaddr,
unsigned long siz

   /* sanity checks */
   if (task->mm == NULL || size == 0UL || vaddr == NULL) {
-   printk(KERN_ERR "perfmon: pfm_remove_smpl_mapping [%d] invalid context mm=%p\n",
task->pid, task->mm);
+   printk(KERN_ERR "perfmon: pfm_remove_smpl_mapping [%d] invalid context mm=%p\n",
task_pid_nr(task), task->mm);
    return -EINVAL;
   }

@@ -1459,7 +1460,7 @@ pfm_remove_smpl_mapping(struct task_struct *task, void *vaddr,
unsigned long siz

   up_write(&task->mm->mmap_sem);
   if (r !=0) {
-   printk(KERN_ERR "perfmon: [%d] unable to unmap sampling buffer @%p size=%lu\n",
task->pid, vaddr, size);
+   printk(KERN_ERR "perfmon: [%d] unable to unmap sampling buffer @%p size=%lu\n",
task_pid_nr(task), vaddr, size);
   }

   DPRINT(("do_unmap(%p, %lu)=%d\n", vaddr, size, r));
@@ -1501,7 +1502,7 @@ pfm_free_smpl_buffer(pfm_context_t *ctx)
   return 0;
```

```
 invalid_free:
- printk(KERN_ERR "perfmon: pfm_free_smpl_buffer [%d] no buffer\n", current->pid);
+ printk(KERN_ERR "perfmon: pfm_free_smpl_buffer [%d] no buffer\n", task_pid_nr(current));
  return -EINVAL;
 }
 #endif
@@ -1554,13 +1555,13 @@ pfm_read(struct file *filp, char __user *buf, size_t size, loff_t *ppos)
  unsigned long flags;
    DECLARE_WAITQUEUE(wait, current);
  if (PFM_IS_FILE(filp) == 0) {
-  printk(KERN_ERR "perfmon: pfm_poll: bad magic [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_poll: bad magic [%d]\n", task_pid_nr(current));
   return -EINVAL;
  }

  ctx = (pfm_context_t *)filp->private_data;
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: pfm_read: NULL ctx [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_read: NULL ctx [%d]\n", task_pid_nr(current));
   return -EINVAL;
  }

@@ -1614,7 +1615,7 @@ pfm_read(struct file *filp, char __user *buf, size_t size, loff_t *ppos)

   PROTECT_CTX(ctx, flags);
  }
- DPRINT(("[%d] back to running ret=%ld\n", current->pid, ret));
+ DPRINT(("[%d] back to running ret=%ld\n", task_pid_nr(current), ret));
    set_current_state(TASK_RUNNING);
  remove_wait_queue(&ctx->ctx_msgq_wait, &wait);

@@ -1623,7 +1624,7 @@ pfm_read(struct file *filp, char __user *buf, size_t size, loff_t *ppos)
  ret = -EINVAL;
  msg = pfm_get_next_msg(ctx);
  if (msg == NULL) {
-  printk(KERN_ERR "perfmon: pfm_read no msg for ctx=%p [%d]\n", ctx, current->pid);
+  printk(KERN_ERR "perfmon: pfm_read no msg for ctx=%p [%d]\n", ctx, task_pid_nr(current));
   goto abort_locked;
  }

@@ -1654,13 +1655,13 @@ pfm_poll(struct file *filp, poll_table * wait)
  unsigned int mask = 0;

  if (PFM_IS_FILE(filp) == 0) {
-  printk(KERN_ERR "perfmon: pfm_poll: bad magic [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_poll: bad magic [%d]\n", task_pid_nr(current));
   return 0;
  }
```

```
  ctx = (pfm_context_t *)filp->private_data;
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: pfm_poll: NULL ctx [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_poll: NULL ctx [%d]\n", task_pid_nr(current));
   return 0;
  }

@@ -1699,7 +1700,7 @@ pfm_do_fasync(int fd, struct file *filp, pfm_context_t *ctx, int on)
  ret = fasync_helper (fd, filp, on, &ctx->ctx_async_queue);

  DPRINT(("pfm_fasync called by [%d] on ctx_fd=%d on=%d async_queue=%p ret=%d\n",
-  current->pid,
+  task_pid_nr(current),
    fd,
    on,
    ctx->ctx_async_queue, ret));
@@ -1714,13 +1715,13 @@ pfm_fasync(int fd, struct file *filp, int on)
  int ret;

  if (PFM_IS_FILE(filp) == 0) {
-  printk(KERN_ERR "perfmon: pfm_fasync bad magic [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_fasync bad magic [%d]\n", task_pid_nr(current));
   return -EBADF;
  }

  ctx = (pfm_context_t *)filp->private_data;
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: pfm_fasync NULL ctx [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_fasync NULL ctx [%d]\n", task_pid_nr(current));
   return -EBADF;
  }
  /*
@@ -1766,7 +1767,7 @@ pfm_syswide_force_stop(void *info)
  if (owner != ctx->ctx_task) {
   printk(KERN_ERR "perfmon: pfm_syswide_force_stop CPU%d unexpected owner [%d] instead
of [%d]\n",
    smp_processor_id(),
-   owner->pid, ctx->ctx_task->pid);
+   task_pid_nr(owner), task_pid_nr(ctx->ctx_task));
   return;
  }
  if (GET_PMU_CTX() != ctx) {
@@ -1776,7 +1777,7 @@ pfm_syswide_force_stop(void *info)
   return;
  }

- DPRINT(("on CPU%d forcing system wide stop for [%d]\n", smp_processor_id(),
```

```
ctx->ctx_task->pid));
+ DPRINT(("on CPU%d forcing system wide stop for [%d]\n", smp_processor_id(),
task_pid_nr(ctx->ctx_task)));
  /*
   * the context is already protected in pfm_close(), we simply
   * need to mask interrupts to avoid a PMU interrupt race on
@@ -1828,7 +1829,7 @@ pfm_flush(struct file *filp, fl_owner_t id)

  ctx = (pfm_context_t *)filp->private_data;
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: pfm_flush: NULL ctx [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_flush: NULL ctx [%d]\n", task_pid_nr(current));
   return -EBADF;
  }

@@ -1976,7 +1977,7 @@ pfm_close(struct inode *inode, struct file *filp)

  ctx = (pfm_context_t *)filp->private_data;
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: pfm_close: NULL ctx [%d]\n", current->pid);
+  printk(KERN_ERR "perfmon: pfm_close: NULL ctx [%d]\n", task_pid_nr(current));
   return -EBADF;
  }

@@ -2073,7 +2074,7 @@ pfm_close(struct inode *inode, struct file *filp)
     */
   ctx->ctx_state = PFM_CTX_ZOMBIE;

-  DPRINT(("zombie ctx for [%d]\n", task->pid));
+  DPRINT(("zombie ctx for [%d]\n", task_pid_nr(task)));
   /*
    * cannot free the context on the spot. deferred until
    * the task notices the ZOMBIE state
@@ -2479,7 +2480,7 @@ pfm_setup_buffer_fmt(struct task_struct *task, struct file *filp,
pfm_context_t
  /* invoke and lock buffer format, if found */
  fmt = pfm_find_buffer_fmt(arg->ctx_smpl_buf_id);
  if (fmt == NULL) {
-  DPRINT(("[%d] cannot find buffer format\n", task->pid));
+  DPRINT(("[%d] cannot find buffer format\n", task_pid_nr(task)));
   return -EINVAL;
  }

@@ -2490,7 +2491,7 @@ pfm_setup_buffer_fmt(struct task_struct *task, struct file *filp,
pfm_context_t

  ret = pfm_buf_fmt_validate(fmt, task, ctx_flags, cpu, fmt_arg);
```

```
- DPRINT(("[%d] after validate(0x%x,%d,%p)=%d\n", task->pid, ctx_flags, cpu, fmt_arg, ret));
+ DPRINT(("[%d] after validate(0x%x,%d,%p)=%d\n", task_pid_nr(task), ctx_flags, cpu, fmt_arg,
ret));

  if (ret) goto error;

@@ -2612,23 +2613,23 @@ pfm_task_incompatible(pfm_context_t *ctx, struct task_struct *task)
   * no kernel task or task not owner by caller
   */
  if (task->mm == NULL) {
-  DPRINT(("task [%d] has not memory context (kernel thread)\n", task->pid));
+  DPRINT(("task [%d] has not memory context (kernel thread)\n", task_pid_nr(task)));
   return -EPERM;
  }
  if (pfm_bad_permissions(task)) {
-  DPRINT(("no permission to attach to  [%d]\n", task->pid));
+  DPRINT(("no permission to attach to  [%d]\n", task_pid_nr(task)));
   return -EPERM;
  }
  /*
   * cannot block in self-monitoring mode
   */
  if (CTX_OVFL_NOBLOCK(ctx) == 0 && task == current) {
-  DPRINT(("cannot load a blocking context on self for [%d]\n", task->pid));
+  DPRINT(("cannot load a blocking context on self for [%d]\n", task_pid_nr(task)));
   return -EINVAL;
  }

  if (task->exit_state == EXIT_ZOMBIE) {
-  DPRINT(("cannot attach to  zombie task [%d]\n", task->pid));
+  DPRINT(("cannot attach to  zombie task [%d]\n", task_pid_nr(task)));
   return -EBUSY;
  }

@@ -2638,7 +2639,7 @@ pfm_task_incompatible(pfm_context_t *ctx, struct task_struct *task)
  if (task == current) return 0;

  if ((task->state != TASK_STOPPED) && (task->state != TASK_TRACED)) {
-  DPRINT(("cannot attach to non-stopped task [%d] state=%ld\n", task->pid, task->state));
+  DPRINT(("cannot attach to non-stopped task [%d] state=%ld\n", task_pid_nr(task),
task->state));
   return -EBUSY;
  }
  /*
@@ -3519,7 +3520,7 @@ pfm_use_debug_registers(struct task_struct *task)

  if (pmu_conf->use_rr_dbregs == 0) return 0;
```

```
- DPRINT(("called for [%d]\n", task->pid));
+ DPRINT(("called for [%d]\n", task_pid_nr(task)));

  /*
   * do it only once
@@ -3550,7 +3551,7 @@ pfm_use_debug_registers(struct task_struct *task)
  DPRINT(("ptrace_use_dbregs=%u  sys_use_dbregs=%u by [%d] ret = %d\n",
     pfm_sessions.pfs_ptrace_use_dbregs,
     pfm_sessions.pfs_sys_use_dbregs,
-    task->pid, ret));
+    task_pid_nr(task), ret));

  UNLOCK_PFS(flags);

@@ -3575,7 +3576,7 @@ pfm_release_debug_registers(struct task_struct *task)

  LOCK_PFS(flags);
  if (pfm_sessions.pfs_ptrace_use_dbregs == 0) {
-  printk(KERN_ERR "perfmon: invalid release for [%d] ptrace_use_dbregs=0\n", task->pid);
+  printk(KERN_ERR "perfmon: invalid release for [%d] ptrace_use_dbregs=0\n",
task_pid_nr(task));
   ret = -1;
  }  else {
   pfm_sessions.pfs_ptrace_use_dbregs--;
@@ -3627,7 +3628,7 @@ pfm_restart(pfm_context_t *ctx, void *arg, int count, struct pt_regs
*regs)

  /* sanity check */
  if (unlikely(task == NULL)) {
-  printk(KERN_ERR "perfmon: [%d] pfm_restart no task\n", current->pid);
+  printk(KERN_ERR "perfmon: [%d] pfm_restart no task\n", task_pid_nr(current));
   return -EINVAL;
  }

@@ -3636,7 +3637,7 @@ pfm_restart(pfm_context_t *ctx, void *arg, int count, struct pt_regs
*regs)
   fmt = ctx->ctx_buf_fmt;

   DPRINT(("restarting self %d ovfl=0x%lx\n",
-    task->pid,
+    task_pid_nr(task),
     ctx->ctx_ovfl_regs[0]));

   if (CTX_HAS_SMPL(ctx)) {
@@ -3660,11 +3661,11 @@ pfm_restart(pfm_context_t *ctx, void *arg, int count, struct pt_regs
*regs)
     pfm_reset_regs(ctx, ctx->ctx_ovfl_regs, PFM_PMD_LONG_RESET);
```

```
    if (rst_ctrl.bits.mask_monitoring == 0) {
-    DPRINT(("resuming monitoring for [%d]\n", task->pid));
+    DPRINT(("resuming monitoring for [%d]\n", task_pid_nr(task)));

     if (state == PFM_CTX_MASKED) pfm_restore_monitoring(task);
    } else {
-    DPRINT(("keeping monitoring stopped for [%d]\n", task->pid));
+    DPRINT(("keeping monitoring stopped for [%d]\n", task_pid_nr(task)));

     // cannot use pfm_stop_monitoring(task, regs);
    }
@@ -3721,10 +3722,10 @@ pfm_restart(pfm_context_t *ctx, void *arg, int count, struct pt_regs
*regs)
   * "self-monitoring".
   */
  if (CTX_OVFL_NOBLOCK(ctx) == 0 && state == PFM_CTX_MASKED) {
-  DPRINT(("unblocking [%d] \n", task->pid));
+  DPRINT(("unblocking [%d] \n", task_pid_nr(task)));
   complete(&ctx->ctx_restart_done);
  } else {
-  DPRINT(("[%d] armed exit trap\n", task->pid));
+  DPRINT(("[%d] armed exit trap\n", task_pid_nr(task)));

   ctx->ctx_fl_trap_reason = PFM_TRAP_REASON_RESET;

@@ -3812,7 +3813,7 @@ pfm_write_ibr_dbr(int mode, pfm_context_t *ctx, void *arg, int count,
struct pt_
   * don't bother if we are loaded and task is being debugged
   */
  if (is_loaded && (thread->flags & IA64_THREAD_DBG_VALID) != 0) {
-  DPRINT(("debug registers already in use for [%d]\n", task->pid));
+  DPRINT(("debug registers already in use for [%d]\n", task_pid_nr(task)));
   return -EBUSY;
  }

@@ -3853,7 +3854,7 @@ pfm_write_ibr_dbr(int mode, pfm_context_t *ctx, void *arg, int count,
struct pt_
   * is shared by all processes running on it
    */
  if (first_time && can_access_pmu) {
-  DPRINT(("[%d] clearing ibrs, dbrs\n", task->pid));
+  DPRINT(("[%d] clearing ibrs, dbrs\n", task_pid_nr(task)));
   for (i=0; i < pmu_conf->num_ibrs; i++) {
    ia64_set_ibr(i, 0UL);
    ia64_dv_serialize_instruction();
@@ -4042,7 +4043,7 @@ pfm_stop(pfm_context_t *ctx, void *arg, int count, struct pt_regs *regs)
   return -EBUSY;
  }
```

```
  DPRINT(("task [%d] ctx_state=%d is_system=%d\n",
-  PFM_CTX_TASK(ctx)->pid,
+  task_pid_nr(PFM_CTX_TASK(ctx)),
   state,
   is_system));
  /*
@@ -4100,7 +4101,7 @@ pfm_stop(pfm_context_t *ctx, void *arg, int count, struct pt_regs *regs)
    * monitoring disabled in kernel at next reschedule
    */
   ctx->ctx_saved_psr_up = 0;
-  DPRINT(("task=[%d]\n", task->pid));
+  DPRINT(("task=[%d]\n", task_pid_nr(task)));
  }
  return 0;
 }
@@ -4305,11 +4306,12 @@ pfm_context_load(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *regs)

   if (is_system) {
    if (pfm_sessions.pfs_ptrace_use_dbregs) {
-    DPRINT(("cannot load [%d] dbregs in use\n", task->pid));
+    DPRINT(("cannot load [%d] dbregs in use\n",
+        task_pid_nr(task)));
     ret = -EBUSY;
    } else {
     pfm_sessions.pfs_sys_use_dbregs++;
-    DPRINT(("load [%d] increased sys_use_dbreg=%u\n", task->pid,
pfm_sessions.pfs_sys_use_dbregs));
+    DPRINT(("load [%d] increased sys_use_dbreg=%u\n", task_pid_nr(task),
pfm_sessions.pfs_sys_use_dbregs));
     set_dbregs = 1;
    }
   }
@@ -4401,7 +4403,7 @@ pfm_context_load(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *regs)

    /* allow user level control */
    ia64_psr(regs)->sp = 0;
-   DPRINT(("clearing psr.sp for [%d]\n", task->pid));
+   DPRINT(("clearing psr.sp for [%d]\n", task_pid_nr(task)));

    SET_LAST_CPU(ctx, smp_processor_id());
    INC_ACTIVATION();
@@ -4436,7 +4438,7 @@ pfm_context_load(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *regs)
    */
   SET_PMU_OWNER(task, ctx);
```

```
-  DPRINT(("context loaded on PMU for [%d]\n", task->pid));
+  DPRINT(("context loaded on PMU for [%d]\n", task_pid_nr(task)));
  } else {
   /*
    * when not current, task MUST be stopped, so this is safe
@@ -4500,7 +4502,7 @@ pfm_context_unload(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *reg
  int prev_state, is_system;
  int ret;

- DPRINT(("ctx_state=%d task [%d]\n", ctx->ctx_state, task ? task->pid : -1));
+ DPRINT(("ctx_state=%d task [%d]\n", ctx->ctx_state, task ? task_pid_nr(task) : -1));

  prev_state = ctx->ctx_state;
  is_system  = ctx->ctx_fl_system;
@@ -4575,7 +4577,7 @@ pfm_context_unload(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *reg
    */
   ia64_psr(regs)->sp = 1;

-  DPRINT(("setting psr.sp for [%d]\n", task->pid));
+  DPRINT(("setting psr.sp for [%d]\n", task_pid_nr(task)));
  }
  /*
   * save PMDs to context
@@ -4615,7 +4617,7 @@ pfm_context_unload(pfm_context_t *ctx, void *arg, int count, struct
pt_regs *reg
  ctx->ctx_fl_can_restart  = 0;
  ctx->ctx_fl_going_zombie = 0;

- DPRINT(("disconnected [%d] from context\n", task->pid));
+ DPRINT(("disconnected [%d] from context\n", task_pid_nr(task)));

  return 0;
 }
@@ -4638,7 +4640,7 @@ pfm_exit_thread(struct task_struct *task)

  PROTECT_CTX(ctx, flags);

- DPRINT(("state=%d task [%d]\n", ctx->ctx_state, task->pid));
+ DPRINT(("state=%d task [%d]\n", ctx->ctx_state, task_pid_nr(task)));

  state = ctx->ctx_state;
  switch(state) {
@@ -4647,13 +4649,13 @@ pfm_exit_thread(struct task_struct *task)
      * only comes to this function if pfm_context is not NULL, i.e., cannot
     * be in unloaded state
      */
```

```
-   printk(KERN_ERR "perfmon: pfm_exit_thread [%d] ctx unloaded\n", task->pid);
+   printk(KERN_ERR "perfmon: pfm_exit_thread [%d] ctx unloaded\n", task_pid_nr(task));
    break;
   case PFM_CTX_LOADED:
   case PFM_CTX_MASKED:
    ret = pfm_context_unload(ctx, NULL, 0, regs);
    if (ret) {
-    printk(KERN_ERR "perfmon: pfm_exit_thread [%d] state=%d unload failed %d\n", task->pid,
state, ret);
+    printk(KERN_ERR "perfmon: pfm_exit_thread [%d] state=%d unload failed %d\n",
task_pid_nr(task), state, ret);
    }
    DPRINT(("ctx unloaded for current state was %d\n", state));

@@ -4662,12 +4664,12 @@ pfm_exit_thread(struct task_struct *task)
   case PFM_CTX_ZOMBIE:
    ret = pfm_context_unload(ctx, NULL, 0, regs);
    if (ret) {
-    printk(KERN_ERR "perfmon: pfm_exit_thread [%d] state=%d unload failed %d\n", task->pid,
state, ret);
+    printk(KERN_ERR "perfmon: pfm_exit_thread [%d] state=%d unload failed %d\n",
task_pid_nr(task), state, ret);
    }
    free_ok = 1;
    break;
   default:
-   printk(KERN_ERR "perfmon: pfm_exit_thread [%d] unexpected state=%d\n", task->pid, state);
+   printk(KERN_ERR "perfmon: pfm_exit_thread [%d] unexpected state=%d\n",
task_pid_nr(task), state);
    break;
  }
  UNPROTECT_CTX(ctx, flags);
@@ -4751,7 +4753,7 @@ recheck:
  DPRINT(("context %d state=%d [%d] task_state=%ld must_stop=%d\n",
   ctx->ctx_fd,
   state,
-  task->pid,
+  task_pid_nr(task),
   task->state, PFM_CMD_STOPPED(cmd)));

  /*
@@ -4798,7 +4800,7 @@ recheck:
   */
  if (PFM_CMD_STOPPED(cmd)) {
   if ((task->state != TASK_STOPPED) && (task->state != TASK_TRACED)) {
-   DPRINT(("[%d] task not in stopped state\n", task->pid));
+   DPRINT(("[%d] task not in stopped state\n", task_pid_nr(task)));
    return -EBUSY;
```

```
   }
   /*
@@ -4891,7 +4893,7 @@ restart_args:
   * limit abuse to min page size
   */
  if (unlikely(sz > PFM_MAX_ARGSIZE)) {
-  printk(KERN_ERR "perfmon: [%d] argument too big %lu\n", current->pid, sz);
+  printk(KERN_ERR "perfmon: [%d] argument too big %lu\n", task_pid_nr(current), sz);
   return -E2BIG;
  }

@@ -5038,11 +5040,11 @@ pfm_context_force_terminate(pfm_context_t *ctx, struct pt_regs
*regs)
 {
  int ret;

- DPRINT(("entering for [%d]\n", current->pid));
+ DPRINT(("entering for [%d]\n", task_pid_nr(current)));

  ret = pfm_context_unload(ctx, NULL, 0, regs);
  if (ret) {
-  printk(KERN_ERR "pfm_context_force_terminate: [%d] unloaded failed with %d\n",
current->pid, ret);
+  printk(KERN_ERR "pfm_context_force_terminate: [%d] unloaded failed with %d\n",
task_pid_nr(current), ret);
  }

  /*
@@ -5079,7 +5081,7 @@ pfm_handle_work(void)

  ctx = PFM_GET_CTX(current);
  if (ctx == NULL) {
-  printk(KERN_ERR "perfmon: [%d] has no PFM context\n", current->pid);
+  printk(KERN_ERR "perfmon: [%d] has no PFM context\n", task_pid_nr(current));
   return;
  }

@@ -5276,7 +5278,7 @@ pfm_overflow_handler(struct task_struct *task, pfm_context_t *ctx, u64
pmc0, str
  DPRINT_ovfl(("pmc0=0x%lx pid=%d iip=0x%lx, %s "
        "used_pmds=0x%lx\n",
    pmc0,
-   task ? task->pid: -1,
+   task ? task_pid_nr(task): -1,
    (regs ? regs->cr_iip : 0),
    CTX_OVFL_NOBLOCK(ctx) ? "nonblocking" : "blocking",
    ctx->ctx_used_pmds[0]));
@@ -5465,7 +5467,7 @@ pfm_overflow_handler(struct task_struct *task, pfm_context_t *ctx, u64
```

```
pmc0, str
  }

  DPRINT_ovfl(("owner [%d] pending=%ld reason=%u ovfl_pmds=0x%lx ovfl_notify=0x%lx
masked=%d\n",
-   GET_PMU_OWNER() ? GET_PMU_OWNER()->pid : -1,
+   GET_PMU_OWNER() ? task_pid_nr(GET_PMU_OWNER()) : -1,
    PFM_GET_WORK_PENDING(task),
    ctx->ctx_fl_trap_reason,
    ovfl_pmds,
@@ -5490,7 +5492,7 @@ pfm_overflow_handler(struct task_struct *task, pfm_context_t *ctx, u64
pmc0, str
 sanity_check:
  printk(KERN_ERR "perfmon: CPU%d overflow handler [%d] pmc0=0x%lx\n",
    smp_processor_id(),
-   task ? task->pid : -1,
+   task ? task_pid_nr(task) : -1,
    pmc0);
  return;

@@ -5523,7 +5525,7 @@ stop_monitoring:
   *
   * Overall pretty hairy stuff....
   */
- DPRINT(("ctx is zombie for [%d], converted to spurious\n", task ? task->pid: -1));
+ DPRINT(("ctx is zombie for [%d], converted to spurious\n", task ? task_pid_nr(task): -1));
  pfm_clear_psr_up();
  ia64_psr(regs)->up = 0;
  ia64_psr(regs)->sp = 1;
@@ -5584,13 +5586,13 @@ pfm_do_interrupt_handler(int irq, void *arg, struct pt_regs *regs)

 report_spurious1:
  printk(KERN_INFO "perfmon: spurious overflow interrupt on CPU%d: process %d has no PFM
context\n",
-  this_cpu, task->pid);
+  this_cpu, task_pid_nr(task));
  pfm_unfreeze_pmu();
  return -1;
 report_spurious2:
  printk(KERN_INFO "perfmon: spurious overflow interrupt on CPU%d: process %d, invalid flag\n",

   this_cpu,
-  task->pid);
+  task_pid_nr(task));
  pfm_unfreeze_pmu();
  return -1;
 }
@@ -5877,7 +5879,8 @@ pfm_force_cleanup(pfm_context_t *ctx, struct pt_regs *regs)
```

```
  ia64_psr(regs)->sp = 1;

  if (GET_PMU_OWNER() == task) {
-  DPRINT(("cleared ownership for [%d]\n", ctx->ctx_task->pid));
+  DPRINT(("cleared ownership for [%d]\n",
+      task_pid_nr(ctx->ctx_task)));
   SET_PMU_OWNER(NULL, NULL);
  }

@@ -5889,7 +5892,7 @@ pfm_force_cleanup(pfm_context_t *ctx, struct pt_regs *regs)
  task->thread.pfm_context  = NULL;
  task->thread.flags       &= ~IA64_THREAD_PM_VALID;

- DPRINT(("force cleanup for [%d]\n",  task->pid));
+ DPRINT(("force cleanup for [%d]\n",  task_pid_nr(task)));
 }


@@ -6433,7 +6436,7 @@ pfm_flush_pmds(struct task_struct *task, pfm_context_t *ctx)

   if (PMD_IS_COUNTING(i)) {
    DPRINT(("[%d] pmd[%d] ctx_pmd=0x%lx hw_pmd=0x%lx\n",
-    task->pid,
+    task_pid_nr(task),
     i,
     ctx->ctx_pmds[i].val,
     val & ovfl_val));
@@ -6455,11 +6458,11 @@ pfm_flush_pmds(struct task_struct *task, pfm_context_t *ctx)
     */
    if (pmc0 & (1UL << i)) {
     val += 1 + ovfl_val;
-    DPRINT(("[%d] pmd[%d] overflowed\n", task->pid, i));
+    DPRINT(("[%d] pmd[%d] overflowed\n", task_pid_nr(task), i));
    }
   }

-  DPRINT(("[%d] ctx_pmd[%d]=0x%lx  pmd_val=0x%lx\n", task->pid, i, val, pmd_val));
+  DPRINT(("[%d] ctx_pmd[%d]=0x%lx  pmd_val=0x%lx\n", task_pid_nr(task), i, val, pmd_val));

   if (is_self) ctx->th_pmds[i] = pmd_val;

@@ -6800,14 +6803,14 @@ dump_pmu_state(const char *from)
  printk("CPU%d from %s() current [%d] iip=0x%lx %s\n",
   this_cpu,
   from,
-  current->pid,
+  task_pid_nr(current),
   regs->cr_iip,
```

```
   current->comm);

  task = GET_PMU_OWNER();
  ctx  = GET_PMU_CTX();

- printk("->CPU%d owner [%d] ctx=%p\n", this_cpu, task ? task->pid : -1, ctx);
+ printk("->CPU%d owner [%d] ctx=%p\n", this_cpu, task ? task_pid_nr(task) : -1, ctx);

  psr = pfm_get_psr();

@@ -6855,7 +6858,7 @@ pfm_inherit(struct task_struct *task, struct pt_regs *regs)
 {
  struct thread_struct *thread;

- DPRINT(("perfmon: pfm_inherit clearing state for [%d]\n", task->pid));
+ DPRINT(("perfmon: pfm_inherit clearing state for [%d]\n", task_pid_nr(task)));

  thread = &task->thread;

diff --git a/arch/ia64/kernel/perfmon_default_smpl.c b/arch/ia64/kernel/perfmon_default_smpl.c
--- a/arch/ia64/kernel/perfmon_default_smpl.c
+++ b/arch/ia64/kernel/perfmon_default_smpl.c
@@ -44,11 +44,11 @@ default_validate(struct task_struct *task, unsigned int flags, int cpu, void
*da
  int ret = 0;

  if (data == NULL) {
-  DPRINT(("[%d] no argument passed\n", task->pid));
+  DPRINT(("[%d] no argument passed\n", task_pid_nr(task)));
   return -EINVAL;
  }

- DPRINT(("[%d] validate flags=0x%x CPU%d\n", task->pid, flags, cpu));
+ DPRINT(("[%d] validate flags=0x%x CPU%d\n", task_pid_nr(task), flags, cpu));

  /*
   * must hold at least the buffer header + one minimally sized entry
@@ -88,7 +88,7 @@ default_init(struct task_struct *task, void *buf, unsigned int flags, int cpu, v
  hdr->hdr_count        = 0UL;

  DPRINT(("[%d] buffer=%p buf_size=%lu hdr_size=%lu hdr_version=%u cur_offs=%lu\n",
-  task->pid,
+  task_pid_nr(task),
   buf,
   hdr->hdr_buf_size,
   sizeof(*hdr),
@@ -245,7 +245,7 @@ default_restart(struct task_struct *task, pfm_ovfl_ctrl_t *ctrl, void *buf,
stru
```

```
 static int
 default_exit(struct task_struct *task, void *buf, struct pt_regs *regs)
 {
- DPRINT(("[%d] exit(%p)\n", task->pid, buf));
+ DPRINT(("[%d] exit(%p)\n", task_pid_nr(task), buf));
  return 0;
 }

diff --git a/arch/ia64/kernel/process.c b/arch/ia64/kernel/process.c
--- a/arch/ia64/kernel/process.c
+++ b/arch/ia64/kernel/process.c
@@ -105,7 +105,8 @@ show_regs (struct pt_regs *regs)
  unsigned long ip = regs->cr_iip + ia64_psr(regs)->ri;

  print_modules();
- printk("\nPid: %d, CPU %d, comm: %20s\n", current->pid, smp_processor_id(), current->comm);
+ printk("\nPid: %d, CPU %d, comm: %20s\n", task_pid_nr(current),
+   smp_processor_id(), current->comm);
  printk("psr : %016lx ifs : %016lx ip  : [<%016lx>]    %s\n",
         regs->cr_ipsr, regs->cr_ifs, ip, print_tainted());
  print_symbol("ip is at %s\n", ip);
diff --git a/arch/ia64/kernel/traps.c b/arch/ia64/kernel/traps.c
--- a/arch/ia64/kernel/traps.c
+++ b/arch/ia64/kernel/traps.c
@@ -61,7 +61,7 @@ die (const char *str, struct pt_regs *regs, long err)

  if (++die.lock_owner_depth < 3) {
   printk("%s[%d]: %s %ld [%d]\n",
-   current->comm, current->pid, str, err, ++die_counter);
+  current->comm, task_pid_nr(current), str, err, ++die_counter);
   (void) notify_die(DIE_OOPS, (char *)str, regs, err, 255, SIGSEGV);
   show_regs(regs);
    } else
@@ -319,7 +319,7 @@ handle_fpu_swa (int fp_fault, struct pt_regs *regs, unsigned long isr)
     last.time = current_jiffies + 5 * HZ;
     printk(KERN_WARNING
            "%s(%d): floating-point assist fault at ip %016lx, isr %016lx\n",
-           current->comm, current->pid, regs->cr_iip + ia64_psr(regs)->ri, isr);
+           current->comm, task_pid_nr(current), regs->cr_iip + ia64_psr(regs)->ri, isr);
    }
   }
  }
@@ -457,7 +457,7 @@ ia64_fault (unsigned long vector, unsigned long isr, unsigned long ifa,
   if (code == 8) {
 # ifdef CONFIG_IA64_PRINT_HAZARDS
    printk("%s[%d]: possible hazard @ ip=%016lx (pr = %016lx)\n",
-          current->comm, current->pid,
+          current->comm, task_pid_nr(current),
```

```
          regs.cr_iip + ia64_psr(&regs)->ri, regs.pr);
 # endif
    return;
diff --git a/arch/ia64/kernel/unaligned.c b/arch/ia64/kernel/unaligned.c
--- a/arch/ia64/kernel/unaligned.c
+++ b/arch/ia64/kernel/unaligned.c
@@ -1340,7 +1340,8 @@ ia64_handle_unaligned (unsigned long ifa, struct pt_regs *regs)
    size_t len;

    len = sprintf(buf, "%s(%d): unaligned access to 0x%016lx, "
-          "ip=0x%016lx\n\r", current->comm, current->pid,
+          "ip=0x%016lx\n\r", current->comm,
+          task_pid_nr(current),
          ifa, regs->cr_iip + ipsr->ri);
    /*
     * Don't call tty_write_message() if we're in the kernel; we might
@@ -1363,7 +1364,7 @@ ia64_handle_unaligned (unsigned long ifa, struct pt_regs *regs)
            "administrator\n"
            "echo 0 > /proc/sys/kernel/ignore-"
            "unaligned-usertrap to re-enable\n",
-           current->comm, current->pid);
+           current->comm, task_pid_nr(current));
    }
   }
  } else {
diff --git a/arch/m32r/kernel/traps.c b/arch/m32r/kernel/traps.c
--- a/arch/m32r/kernel/traps.c
+++ b/arch/m32r/kernel/traps.c
@@ -196,7 +196,7 @@ static void show_registers(struct pt_regs *regs)
   printk("SPI: %08lx\n", sp);
  }
  printk("Process %s (pid: %d, process nr: %d, stackpage=%08lx)",
-  current->comm, current->pid, 0xffff & i, 4096+(unsigned long)current);
+  current->comm, task_pid_nr(current), 0xffff & i, 4096+(unsigned long)current);

  /*
   * When in-kernel, we also print out the stack and code at the
diff --git a/arch/m68k/kernel/traps.c b/arch/m68k/kernel/traps.c
--- a/arch/m68k/kernel/traps.c
+++ b/arch/m68k/kernel/traps.c
@@ -900,7 +900,7 @@ void show_registers(struct pt_regs *regs)
         regs->d4, regs->d5, regs->a0, regs->a1);

  printk("Process %s (pid: %d, task=%p)\n",
-  current->comm, current->pid, current);
+  current->comm, task_pid_nr(current), current);
  addr = (unsigned long)&fp->un;
  printk("Frame format=%X ", regs->format);
```

```
  switch (regs->format) {
@@ -1038,7 +1038,7 @@ void bad_super_trap (struct frame *fp)
     fp->un.fmtb.daddr, space_names[ssw & DFC],
     fp->ptregs.pc);
  }
- printk ("Current process id is %d\n", current->pid);
+ printk ("Current process id is %d\n", task_pid_nr(current));
  die_if_kernel("BAD KERNEL TRAP", &fp->ptregs, 0);
 }

diff --git a/arch/mips/kernel/traps.c b/arch/mips/kernel/traps.c
--- a/arch/mips/kernel/traps.c
+++ b/arch/mips/kernel/traps.c
@@ -307,7 +307,7 @@ void show_registers(struct pt_regs *regs)
  show_regs(regs);
  print_modules();
  printk("Process %s (pid: %d, threadinfo=%p, task=%p)\n",
-        current->comm, current->pid, current_thread_info(), current);
+         current->comm, task_pid_nr(current), current_thread_info(), current);
  show_stacktrace(current, regs);
  show_code((unsigned int __user *) regs->cp0_epc);
  printk("\n");
diff --git a/arch/parisc/kernel/traps.c b/arch/parisc/kernel/traps.c
--- a/arch/parisc/kernel/traps.c
+++ b/arch/parisc/kernel/traps.c
@@ -219,7 +219,7 @@ void die_if_kernel(char *str, struct pt_regs *regs, long err)
    return; /* STFU */

   printk(KERN_CRIT "%s (pid %d): %s (code %ld) at " RFMT "\n",
-   current->comm, current->pid, str, err, regs->iaoq[0]);
+   current->comm, task_pid_nr(current), str, err, regs->iaoq[0]);
 #ifdef PRINT_USER_FAULTS
   /* XXX for debugging only */
   show_regs(regs);
@@ -252,7 +252,7 @@ KERN_CRIT "                    ||     ||\n");

  if (err)
   printk(KERN_CRIT "%s (pid %d): %s (code %ld)\n",
-   current->comm, current->pid, str, err);
+   current->comm, task_pid_nr(current), str, err);

  /* Wot's wrong wif bein' racy? */
  if (current->thread.flags & PARISC_KERNEL_DEATH) {
@@ -317,7 +317,7 @@ static void handle_break(struct pt_regs *regs)
  if (unlikely(iir != GDB_BREAK_INSN)) {
   printk(KERN_DEBUG "break %d,%d: pid=%d command='%s'\n",
    iir & 31, (iir>>13) & ((1<<13)-1),
-   current->pid, current->comm);
```

```
+   task_pid_nr(current), current->comm);
   show_regs(regs);
  }
 #endif
@@ -747,7 +747,7 @@ void handle_interruption(int code, struct pt_regs *regs)
   if (user_mode(regs)) {
 #ifdef PRINT_USER_FAULTS
    printk(KERN_DEBUG "\nhandle_interruption() pid=%d command='%s'\n",
-       current->pid, current->comm);
+       task_pid_nr(current), current->comm);
    show_regs(regs);
 #endif
    /* SIGBUS, for lack of a better one. */
@@ -772,7 +772,7 @@ void handle_interruption(int code, struct pt_regs *regs)
   else
    printk(KERN_DEBUG "User Fault (long pointer) (fault %d) ",
          code);
-  printk("pid=%d command='%s'\n", current->pid, current->comm);
+  printk("pid=%d command='%s'\n", task_pid_nr(current), current->comm);
   show_regs(regs);
 #endif
   si.si_signo = SIGSEGV;
diff --git a/arch/parisc/kernel/unaligned.c b/arch/parisc/kernel/unaligned.c
--- a/arch/parisc/kernel/unaligned.c
+++ b/arch/parisc/kernel/unaligned.c
@@ -469,7 +469,7 @@ void handle_unaligned(struct pt_regs *regs)
       && ++unaligned_count < 5) {
    char buf[256];
    sprintf(buf, "%s(%d): unaligned access to 0x" RFMT " at ip=0x" RFMT "\n",
-    current->comm, current->pid, regs->ior, regs->iaoq[0]);
+    current->comm, task_pid_nr(current), regs->ior, regs->iaoq[0]);
    printk(KERN_WARNING "%s", buf);
 #ifdef DEBUG_UNALIGNED
    show_regs(regs);
diff --git a/arch/parisc/mm/fault.c b/arch/parisc/mm/fault.c
--- a/arch/parisc/mm/fault.c
+++ b/arch/parisc/mm/fault.c
@@ -211,7 +211,7 @@ bad_area:
 #ifdef PRINT_USER_FAULTS
   printk(KERN_DEBUG "\n");
   printk(KERN_DEBUG "do_page_fault() pid=%d command='%s' type=%lu address=0x%08lx\n",
-      tsk->pid, tsk->comm, code, address);
+      task_pid_nr(tsk), tsk->comm, code, address);
   if (vma) {
    printk(KERN_DEBUG "vm_start = 0x%08lx, vm_end = 0x%08lx\n",
      vma->vm_start, vma->vm_end);
diff --git a/arch/powerpc/kernel/process.c b/arch/powerpc/kernel/process.c
--- a/arch/powerpc/kernel/process.c
```

```
+++ b/arch/powerpc/kernel/process.c
@@ -429,7 +429,7 @@ void show_regs(struct pt_regs * regs)
   printk("DAR: "REG", DSISR: "REG"\n", regs->dar, regs->dsisr);
 #endif
  printk("TASK = %p[%d] '%s' THREAD: %p",
-        current, current->pid, current->comm, task_thread_info(current));
+        current, task_pid_nr(current), current->comm, task_thread_info(current));

 #ifdef CONFIG_SMP
  printk(" CPU: %d", smp_processor_id());
diff --git a/arch/powerpc/kernel/traps.c b/arch/powerpc/kernel/traps.c
--- a/arch/powerpc/kernel/traps.c
+++ b/arch/powerpc/kernel/traps.c
@@ -878,7 +878,7 @@ void nonrecoverable_exception(struct pt_regs *regs)
 void trace_syscall(struct pt_regs *regs)
 {
  printk("Task: %p(%d), PC: %08lX/%08lX, Syscall: %3ld, Result: %s%ld    %s\n",
-        current, current->pid, regs->nip, regs->link, regs->gpr[0],
+        current, task_pid_nr(current), regs->nip, regs->link, regs->gpr[0],
        regs->ccr&0x10000000?"Error=":"", regs->gpr[3], print_tainted());
 }

diff --git a/arch/s390/kernel/process.c b/arch/s390/kernel/process.c
--- a/arch/s390/kernel/process.c
+++ b/arch/s390/kernel/process.c
@@ -166,7 +166,7 @@ void show_regs(struct pt_regs *regs)

        printk("CPU:    %d    %s\n", task_thread_info(tsk)->cpu, print_tainted());
        printk("Process %s (pid: %d, task: %p, ksp: %p)\n",
-        current->comm, current->pid, (void *) tsk,
+        current->comm, task_pid_nr(current), (void *) tsk,
        (void *) tsk->thread.ksp);

  show_registers(regs);
diff --git a/arch/sh/kernel/process.c b/arch/sh/kernel/process.c
--- a/arch/sh/kernel/process.c
+++ b/arch/sh/kernel/process.c
@@ -120,7 +120,7 @@ void machine_power_off(void)
 void show_regs(struct pt_regs * regs)
 {
  printk("\n");
- printk("Pid : %d, Comm: %20s\n", current->pid, current->comm);
+ printk("Pid : %d, Comm: %20s\n", task_pid_nr(current), current->comm);
  print_symbol("PC is at %s\n", instruction_pointer(regs));
  printk("PC  : %08lx SP  : %08lx SR  : %08lx ",
        regs->pc, regs->regs[15], regs->sr);
diff --git a/arch/sh/kernel/signal.c b/arch/sh/kernel/signal.c
--- a/arch/sh/kernel/signal.c
```

```
+++ b/arch/sh/kernel/signal.c
@@ -382,7 +382,7 @@ static int setup_frame(int sig, struct k_sigaction *ka,
  set_fs(USER_DS);

  pr_debug("SIG deliver (%s:%d): sp=%p pc=%08lx pr=%08lx\n",
-   current->comm, current->pid, frame, regs->pc, regs->pr);
+   current->comm, task_pid_nr(current), frame, regs->pc, regs->pr);

  flush_cache_sigtramp(regs->pr);

@@ -462,7 +462,7 @@ static int setup_rt_frame(int sig, struct k_sigaction *ka, siginfo_t *info,
  set_fs(USER_DS);

  pr_debug("SIG deliver (%s:%d): sp=%p pc=%08lx pr=%08lx\n",
-   current->comm, current->pid, frame, regs->pc, regs->pr);
+   current->comm, task_pid_nr(current), frame, regs->pc, regs->pr);

  flush_cache_sigtramp(regs->pr);

diff --git a/arch/sh/kernel/traps.c b/arch/sh/kernel/traps.c
--- a/arch/sh/kernel/traps.c
+++ b/arch/sh/kernel/traps.c
@@ -91,8 +91,8 @@ void die(const char * str, struct pt_regs * regs, long err)
  print_modules();
  show_regs(regs);

- printk("Process: %s (pid: %d, stack limit = %p)\n",
-        current->comm, current->pid, task_stack_page(current) + 1);
+ printk("Process: %s (pid: %d, stack limit = %p)\n", current->comm,
+   task_pid_nr(current), task_stack_page(current) + 1);

  if (!user_mode(regs) || in_interrupt())
   dump_mem("Stack: ", regs->regs[15], THREAD_SIZE +
@@ -382,7 +382,8 @@ static int handle_unaligned_access(u16 instruction, struct pt_regs *regs)

   printk(KERN_NOTICE "Fixing up unaligned userspace access "
          "in \"%s\" pid=%d pc=0x%p ins=0x%04hx\n",
-         current->comm,current->pid,(u16*)regs->pc,instruction);
+         current->comm, task_pid_nr(current),
+         (u16 *)regs->pc, instruction);
  }

  ret = -EFAULT;
diff --git a/arch/sh64/kernel/traps.c b/arch/sh64/kernel/traps.c
--- a/arch/sh64/kernel/traps.c
+++ b/arch/sh64/kernel/traps.c
@@ -764,7 +764,7 @@ static int misaligned_fixup(struct pt_regs *regs)
   --user_mode_unaligned_fixup_count;
```

```
   /* Only do 'count' worth of these reports, to remove a potential DoS against syslog */
   printk("Fixing up unaligned userspace access in \"%s\" pid=%d pc=0x%08x ins=0x%08lx\n",
-         current->comm, current->pid, (__u32)regs->pc, opcode);
+         current->comm, task_pid_nr(current), (__u32)regs->pc, opcode);
  } else
 #endif
  if (!user_mode(regs) && (kernel_mode_unaligned_fixup_count > 0)) {
@@ -774,7 +774,7 @@ static int misaligned_fixup(struct pt_regs *regs)
          (__u32)regs->pc, opcode);
   } else {
    printk("Fixing up unaligned kernelspace access in \"%s\" pid=%d pc=0x%08x ins=0x%08lx\n",
-         current->comm, current->pid, (__u32)regs->pc, opcode);
+         current->comm, task_pid_nr(current), (__u32)regs->pc, opcode);
   }
  }

diff --git a/arch/sh64/mm/fault.c b/arch/sh64/mm/fault.c
--- a/arch/sh64/mm/fault.c
+++ b/arch/sh64/mm/fault.c
@@ -81,7 +81,7 @@ static inline void print_vma(struct vm_area_struct *vma)

 static inline void print_task(struct task_struct *tsk)
 {
- printk("Task pid %d\n", tsk->pid);
+ printk("Task pid %d\n", task_pid_nr(tsk));
 }

 static pte_t *lookup_pte(struct mm_struct *mm, unsigned long address)
@@ -272,7 +272,7 @@ bad_area:
     * usermode, so only need a few */
    count++;
    printk("user mode bad_area address=%08lx pid=%d (%s) pc=%08lx\n",
-    address, current->pid, current->comm,
+    address, task_pid_nr(current), current->comm,
     (unsigned long) regs->pc);
 #if 0
    show_regs(regs);
diff --git a/arch/sparc/kernel/ptrace.c b/arch/sparc/kernel/ptrace.c
--- a/arch/sparc/kernel/ptrace.c
+++ b/arch/sparc/kernel/ptrace.c
@@ -155,7 +155,7 @@ static inline void read_sunos_user(struct pt_regs *regs, unsigned long
offset,
   /* Rest of them are completely unsupported. */
  default:
   printk("%s [%d]: Wants to read user offset %ld\n",
-         current->comm, current->pid, offset);
+         current->comm, task_pid_nr(current), offset);
   pt_error_return(regs, EIO);
```

```
   return;
  }
@@ -222,7 +222,7 @@ static inline void write_sunos_user(struct pt_regs *regs, unsigned long
offset,
    /* Rest of them are completely unsupported or "no-touch". */
   default:
    printk("%s [%d]: Wants to write user offset %ld\n",
-          current->comm, current->pid, offset);
+          current->comm, task_pid_nr(current), offset);
    goto failure;
   }
  success:
diff --git a/arch/sparc/kernel/sys_sparc.c b/arch/sparc/kernel/sys_sparc.c
--- a/arch/sparc/kernel/sys_sparc.c
+++ b/arch/sparc/kernel/sys_sparc.c
@@ -357,7 +357,7 @@ c_sys_nis_syscall (struct pt_regs *regs)
   if (count++ > 5)
    return -ENOSYS;
   printk ("%s[%d]: Unimplemented SPARC system call %d\n",
-  current->comm, current->pid, (int)regs->u_regs[1]);
+  current->comm, task_pid_nr(current), (int)regs->u_regs[1]);
  #ifdef DEBUG_UNIMP_SYSCALL
   show_regs (regs);
  #endif
diff --git a/arch/sparc/kernel/traps.c b/arch/sparc/kernel/traps.c
--- a/arch/sparc/kernel/traps.c
+++ b/arch/sparc/kernel/traps.c
@@ -38,7 +38,7 @@ struct trap_trace_entry trapbuf[1024];

  void syscall_trace_entry(struct pt_regs *regs)
  {
- printk("%s[%d]: ", current->comm, current->pid);
+ printk("%s[%d]: ", current->comm, task_pid_nr(current));
   printk("scall<%d> (could be %d)\n", (int) regs->u_regs[UREG_G1],
          (int) regs->u_regs[UREG_I0]);
  }
@@ -99,7 +99,7 @@ void die_if_kernel(char *str, struct pt_regs *regs)
  "              /_| \\__/ |_\\\n"
  "                 \\__U_/\n");

- printk("%s(%d): %s [#%d]\n", current->comm, current->pid, str, ++die_counter);
+ printk("%s(%d): %s [#%d]\n", current->comm, task_pid_nr(current), str, ++die_counter);
   show_regs(regs);
   add_taint(TAINT_DIE);

diff --git a/arch/sparc64/kernel/traps.c b/arch/sparc64/kernel/traps.c
--- a/arch/sparc64/kernel/traps.c
+++ b/arch/sparc64/kernel/traps.c
```

```
@@ -2225,7 +2225,7 @@ void die_if_kernel(char *str, struct pt_regs *regs)
 "             /_| \\__/ |_\\\n"
 "               \\__U_/\n");

- printk("%s(%d): %s [#%d]\n", current->comm, current->pid, str, ++die_counter);
+ printk("%s(%d): %s [#%d]\n", current->comm, task_pid_nr(current), str, ++die_counter);
  notify_die(DIE_OOPS, str, regs, 0, 255, SIGSEGV);
  __asm__ __volatile__("flushw");
  __show_regs(regs);
diff --git a/arch/um/sys-x86_64/sysrq.c b/arch/um/sys-x86_64/sysrq.c
--- a/arch/um/sys-x86_64/sysrq.c
+++ b/arch/um/sys-x86_64/sysrq.c
@@ -15,8 +15,8 @@ void __show_regs(struct pt_regs * regs)
 {
  printk("\n");
  print_modules();
- printk("Pid: %d, comm: %.20s %s %s\n",
-        current->pid, current->comm, print_tainted(), init_utsname()->release);
+ printk("Pid: %d, comm: %.20s %s %s\n", task_pid_nr(current),
+  current->comm, print_tainted(), init_utsname()->release);
  printk("RIP: %04lx:[<%016lx>] ", PT_REGS_CS(regs) & 0xffff,
         PT_REGS_RIP(regs));
  printk("\nRSP: %016lx  EFLAGS: %08lx\n", PT_REGS_RSP(regs),
diff --git a/arch/xtensa/kernel/traps.c b/arch/xtensa/kernel/traps.c
--- a/arch/xtensa/kernel/traps.c
+++ b/arch/xtensa/kernel/traps.c
@@ -176,7 +176,7 @@ void do_unhandled(struct pt_regs *regs, unsigned long exccause)
  printk("Caught unhandled exception in '%s' "
         "(pid = %d, pc = %#010lx) - should not happen\n"
         "\tEXCCAUSE is %ld\n",
-        current->comm, current->pid, regs->pc, exccause);
+        current->comm, task_pid_nr(current), regs->pc, exccause);
  force_sig(SIGILL, current);
 }

@@ -228,7 +228,7 @@ do_illegal_instruction(struct pt_regs *regs)
  /* If in user mode, send SIGILL signal to current process. */

  printk("Illegal Instruction in '%s' (pid = %d, pc = %#010lx)\n",
-     current->comm, current->pid, regs->pc);
+     current->comm, task_pid_nr(current), regs->pc);
  force_sig(SIGILL, current);
 }

@@ -254,7 +254,7 @@ do_unaligned_user (struct pt_regs *regs)
  current->thread.error_code = -3;
  printk("Unaligned memory access to %08lx in '%s' "
         "(pid = %d, pc = %#010lx)\n",
```

```
-       regs->excvaddr, current->comm, current->pid, regs->pc);
+        regs->excvaddr, current->comm, task_pid_nr(current), regs->pc);
  info.si_signo = SIGBUS;
  info.si_errno = 0;
  info.si_code = BUS_ADRALN;
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers