

---

Subject: Re: [RFC][PATCH 06/16] Define is\_global\_init()  
Posted by [serue](#) on Thu, 24 May 2007 15:27:43 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Quoting [sukadev@us.ibm.com](mailto:sukadev@us.ibm.com) ([sukadev@us.ibm.com](mailto:sukadev@us.ibm.com)):

>  
> Subject: Define is\_global\_init()  
>  
> From: Serge E. Hallyn <[serue@us.ibm.com](mailto:serue@us.ibm.com)>  
>  
>  
> is\_init() is an ambiguous name for the pid==1 check. Split it into  
> is\_global\_init() and is\_container\_init().  
>  
> A container init has it's tsk->pid == 1.  
>  
> A global init also has it's tsk->pid == 1, and it's active pid namespace  
> is the init\_pid\_ns.  
>  
> Changelog:  
>  
> 2.6.21-mm2-pidns2:  
>  
> - [Sukadev Bhattiprolu] Changed is\_container\_init() calls in {powerpc,  
> ppc,avr32}/traps.c for the \_\_exception() call to is\_global\_init().  
> This way, we kill only the container if the container's init has a  
> bug rather than force a kernel panic.  
>  
> Signed-off-by: Serge E. Hallyn <[serue@us.ibm.com](mailto:serue@us.ibm.com)>  
> Signed-off-by: Sukadev Bhattiprolu <[sukadev@us.ibm.com](mailto:sukadev@us.ibm.com)>  
> ---  
> arch/alpha/mm/fault.c | 2 +-  
> arch/arm/mm/fault.c | 2 +-  
> arch/arm26/mm/fault.c | 2 +-  
> arch/avr32/kernel/traps.c | 2 +-  
> arch/avr32/mm/fault.c | 6 +++---  
> arch/i386/lib/usercopy.c | 2 +-  
> arch/i386/mm/fault.c | 2 +-  
> arch/ia64/mm/fault.c | 2 +-  
> arch/m32r/mm/fault.c | 2 +-  
> arch/m68k/mm/fault.c | 2 +-  
> arch/mips/mm/fault.c | 2 +-  
> arch/powerpc/kernel/traps.c | 2 +-  
> arch/powerpc/mm/fault.c | 2 +-  
> arch/powerpc/platforms/pseries/ras.c | 2 +-  
> arch/ppc/kernel/traps.c | 2 +-  
> arch/ppc/mm/fault.c | 2 +-  
> arch/s390/lib/uaccess\_pt.c | 2 +-  
>

```

> arch/s390/mm/fault.c          | 2 +-
> arch/sh/mm/fault.c           | 2 +-
> arch/sh64/mm/fault.c         | 6 +++---
> arch/um/kernel/trap.c        | 2 +-
> arch/x86_64/mm/fault.c       | 4 +++--
> arch/xtensa/mm/fault.c       | 2 +-
> drivers/char/sysrq.c         | 2 +-
> include/linux/sched.h        | 13 +++-----
> kernel/capability.c          | 3 +-
> kernel/exit.c                | 2 +-
> kernel/kexec.c               | 2 +-
> kernel/pid.c                 | 21 ++++++
> kernel/sysctl.c              | 2 +-
> mm/oom_kill.c                | 4 +++--
> security/commoncap.c         | 2 +-
> 32 files changed, 61 insertions(+), 46 deletions(-)
>
> Index: lx26-21-mm2/arch/alpha/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/alpha/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/alpha/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -192,7 +192,7 @@ do_page_fault(unsigned long address, uns
> /* We ran out of memory, or some other thing happened to us that
>    made us unable to handle the page fault gracefully. */
> out_of_memory:
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> Index: lx26-21-mm2/arch/arm/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/arm/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/arm/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -197,7 +197,7 @@ survive:
>     return fault;
> }
>
> - if (!is_init(tsk))
> + if (!is_global_init(tsk))
>     goto out;
>
> /*
> Index: lx26-21-mm2/arch/arm26/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/arm26/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/arm26/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -185,7 +185,7 @@ survive:

```

```

> }
>
> fault = -3; /* out of memory */
> - if (!is_init(tsk))
> + if (!is_global_init(tsk))
> goto out;
>
> /*
> Index: lx26-21-mm2/arch/i386/lib/usercopy.c
> =====
> --- lx26-21-mm2.orig/arch/i386/lib/usercopy.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/i386/lib/usercopy.c 2007-05-22 16:59:46.000000000 -0700
> @@ -748,7 +748,7 @@ survive:
>     retval = get_user_pages(current, current->mm,
>         (unsigned long )to, 1, 1, 0, &pg, NULL);
>
> - if (retval == -ENOMEM && is_init(current)) {
> + if (retval == -ENOMEM && is_global_init(current)) {
>     up_read(&current->mm->mmap_sem);
>     congestion_wait(WRITE, HZ/50);
>     goto survive;
> Index: lx26-21-mm2/arch/i386/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/i386/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/i386/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -576,7 +576,7 @@ no_context:
>     */
> out_of_memory:
>     up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> Index: lx26-21-mm2/arch/ia64/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/ia64/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/ia64/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -279,7 +279,7 @@ ia64_do_page_fault (unsigned long address
>
> out_of_memory:
>     up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> Index: lx26-21-mm2/arch/m32r/mm/fault.c

```

```

> =====
> --- lx26-21-mm2.orig/arch/m32r/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/m32r/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -272,7 +272,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-21-mm2/arch/m68k/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/m68k/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/m68k/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -181,7 +181,7 @@ good_area:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-21-mm2/arch/mips/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/mips/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/mips/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -174,7 +174,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-21-mm2/arch/powerpc/kernel/traps.c
> =====
> --- lx26-21-mm2.orig/arch/powerpc/kernel/traps.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/powerpc/kernel/traps.c 2007-05-22 16:59:46.000000000 -0700
> @@ -190,7 +190,7 @@ void _exception(int signr, struct pt_reg
> * generate the same exception over and over again and we get
> * nowhere. Better to kill it and let the kernel panic.
> */
> - if (is_init(current)) {
> + if (is_global_init(current)) {

```

```

> __sighandler_t handler;
>
> spin_lock_irq(&current->sighand->siglock);
> Index: lx26-21-mm2/arch/powerpc/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/powerpc/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/powerpc/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -374,7 +374,7 @@ bad_area_nosemaphore:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> Index: lx26-21-mm2/arch/powerpc/platforms/pseries/ras.c
> =====
> --- lx26-21-mm2.orig/arch/powerpc/platforms/pseries/ras.c 2007-05-22 16:58:38.000000000
-0700
> +++ lx26-21-mm2/arch/powerpc/platforms/pseries/ras.c 2007-05-22 16:59:46.000000000 -0700
> @@ -332,7 +332,7 @@ static int recover_mce(struct pt_regs *r
>     err->disposition == RTAS_DISP_NOT_RECOVERED &&
>     err->target == RTAS_TARGET_MEMORY &&
>     err->type == RTAS_TYPE_ECC_UNCORR &&
> -    !(current->pid == 0 || is_init(current))) {
> +    !(current->pid == 0 || is_global_init(current))) {
>     /* Kill off a user process with an ECC error */
>     printk(KERN_ERR "MCE: uncorrectable ecc error for pid %d\n",
>         current->pid);
> Index: lx26-21-mm2/arch/ppc/kernel/traps.c
> =====
> --- lx26-21-mm2.orig/arch/ppc/kernel/traps.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/ppc/kernel/traps.c 2007-05-22 16:59:46.000000000 -0700
> @@ -120,7 +120,7 @@ void _exception(int signr, struct pt_reg
>     * generate the same exception over and over again and we get
>     * nowhere. Better to kill it and let the kernel panic.
>     */
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     __sighandler_t handler;
>
>     spin_lock_irq(&current->sighand->siglock);
> Index: lx26-21-mm2/arch/ppc/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/ppc/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/ppc/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -291,7 +291,7 @@ bad_area:

```

```

> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-21-mm2/arch/s390/lib/uaccess_pt.c
> =====
> --- lx26-21-mm2.orig/arch/s390/lib/uaccess_pt.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/s390/lib/uaccess_pt.c 2007-05-22 16:59:46.000000000 -0700
> @@ -65,7 +65,7 @@ out:
>
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-21-mm2/arch/s390/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/s390/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/s390/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -211,7 +211,7 @@ static int do_out_of_memory(struct pt_re
>   struct mm_struct *mm = tsk->mm;
>
>   up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   return 1;
> Index: lx26-21-mm2/arch/sh/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/sh/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/sh/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -233,7 +233,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-21-mm2/arch/sh64/mm/fault.c

```

```

> =====
> --- lx26-21-mm2.orig/arch/sh64/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/sh64/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -276,7 +276,7 @@ bad_area:
>   show_regs(regs);
> #endif
> }
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
>   panic("INIT had user mode bad_area\n");
> }
>   tsk->thread.address = address;
> @@ -318,14 +318,14 @@ no_context:
>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   panic("INIT out of memory\n");
>   yield();
>   goto survive;
> }
> printk("fault:Out of memory\n");
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-21-mm2/arch/um/kernel/trap.c
> =====
> --- lx26-21-mm2.orig/arch/um/kernel/trap.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/um/kernel/trap.c 2007-05-22 16:59:46.000000000 -0700
> @@ -120,7 +120,7 @@ out_nosemaphore:
>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   up_read(&mm->mmap_sem);
>   yield();
>   down_read(&mm->mmap_sem);
> Index: lx26-21-mm2/arch/x86_64/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/x86_64/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/x86_64/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -223,7 +223,7 @@ static int is_errata93(struct pt_regs *r
>

```

```

> int unhandled_signal(struct task_struct *tsk, int sig)
> {
> - if (is_init(tsk))
> + if (is_global_init(tsk))
>     return 1;
> if (tsk->ptrace & PT_PTRACED)
>     return 0;
> @@ -557,7 +557,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     goto again;
> }
> Index: lx26-21-mm2/arch/xtensa/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/xtensa/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/xtensa/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -144,7 +144,7 @@ bad_area:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> Index: lx26-21-mm2/drivers/char/sysrq.c
> =====
> --- lx26-21-mm2.orig/drivers/char/sysrq.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/drivers/char/sysrq.c 2007-05-22 16:59:46.000000000 -0700
> @@ -250,7 +250,7 @@ static void send_sig_all(int sig)
> struct task_struct *p;
>
> for_each_process(p) {
> - if (p->mm && !is_init(p))
> + if (p->mm && !is_global_init(p))
>     /* Not swapper, init nor kernel thread */
>     force_sig(sig, p);
> }
> Index: lx26-21-mm2/include/linux/sched.h
> =====
> --- lx26-21-mm2.orig/include/linux/sched.h 2007-05-22 16:59:44.000000000 -0700
> +++ lx26-21-mm2/include/linux/sched.h 2007-05-22 16:59:46.000000000 -0700
> @@ -1171,16 +1171,9 @@ static inline int pid_alive(struct task_
> return p->pids[PIDTYPE_PID].pid != NULL;

```



```

> }
>
> -/**
> - * is_init - check if a task structure is init
> - * @tsk: Task structure to be checked.
> - *
> - * Check if a task structure is the first user space task the kernel created.
> - */
> -static inline int is_init(struct task_struct *tsk)
> -{
> - return tsk->pid == 1;
> -}
> +struct pid_namespace;
> +extern int is_global_init(struct task_struct *tsk);
> +extern int is_container_init(struct task_struct *tsk);
>
> extern struct pid *cad_pid;
>
> Index: lx26-21-mm2/kernel/capability.c
> =====
> --- lx26-21-mm2.orig/kernel/capability.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/kernel/capability.c 2007-05-22 16:59:46.000000000 -0700
> @@ -12,6 +12,7 @@
> #include <linux/module.h>
> #include <linux/security.h>
> #include <linux/syscalls.h>
> +#include <linux/pid_namespace.h>
> #include <asm/uaccess.h>
>
> unsigned securebits = SECUREBITS_DEFAULT; /* systemwide security settings */
> @@ -135,7 +136,7 @@ static inline int cap_set_all(kernel_cap
>     int found = 0;
>
>     do_each_thread(g, target) {
> -         if (target == current || is_init(target))
> +         if (target == current || is_container_init(target))

```

Ok, this is simply conforming to the capset(2) manpage/specs, so that is correct. (as the only actual usage of is\_container\_init()).

```

>         continue;
>         found = 1;
>     if (security_capset_check(target, effective, inheritable,
> Index: lx26-21-mm2/kernel/exit.c
> =====
> --- lx26-21-mm2.orig/kernel/exit.c 2007-05-22 16:59:42.000000000 -0700
> +++ lx26-21-mm2/kernel/exit.c 2007-05-22 16:59:46.000000000 -0700
> @@ -230,7 +230,7 @@ static int will_become_orphaned_pgrp(str

```

```

> do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
>   if (p == ignored_task
>       || p->exit_state
> -   || is_init(p->real_parent))
> +   || is_global_init(p->real_parent))
>     continue;
>   if (task_pgrp(p->real_parent) != pgrp &&
>       task_session(p->real_parent) == task_session(p)) {
> Index: lx26-21-mm2/kernel/kexec.c
> =====
> --- lx26-21-mm2.orig/kernel/kexec.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/kernel/kexec.c 2007-05-22 16:59:46.000000000 -0700
> @@ -42,7 +42,7 @@ struct resource crashk_res = {
>
> int kexec_should_crash(struct task_struct *p)
> {
> - if (in_interrupt() || !p->pid || is_init(p) || panic_on_oops)
> + if (in_interrupt() || !p->pid || is_global_init(p) || panic_on_oops)
>   return 1;
>   return 0;
> }
> Index: lx26-21-mm2/kernel/sysctl.c
> =====
> --- lx26-21-mm2.orig/kernel/sysctl.c 2007-05-22 16:59:41.000000000 -0700
> +++ lx26-21-mm2/kernel/sysctl.c 2007-05-22 16:59:46.000000000 -0700
> @@ -1730,7 +1730,7 @@ int proc_dointvec_bset(ctl_table *table,
>   return -EPERM;
> }
>
> - op = is_init(current) ? OP_SET : OP_AND;
> + op = is_global_init(current) ? OP_SET : OP_AND;
>   return do_proc_dointvec(table, write, filp, buffer, lenp, ppos,
>       do_proc_dointvec_bset_conv, &op);
> }
> Index: lx26-21-mm2/mm/oom_kill.c
> =====
> --- lx26-21-mm2.orig/mm/oom_kill.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/mm/oom_kill.c 2007-05-22 16:59:46.000000000 -0700
> @@ -222,7 +222,7 @@ static struct task_struct *select_bad_pr
>   if (!p->mm)
>     continue;
>   /* skip the init task */
> - if (is_init(p))
> + if (is_global_init(p))
>     continue;
>
>   /*
> @@ -275,7 +275,7 @@ static struct task_struct *select_bad_pr

```

```

> */
> static void __oom_kill_task(struct task_struct *p, int verbose)
> {
> - if (is_init(p)) {
> + if (is_global_init(p)) {
>   WARN_ON(1);
>   printk(KERN_WARNING "tried to kill init!\n");
>   return;
> Index: lx26-21-mm2/security/commoncap.c
> =====
> --- lx26-21-mm2.orig/security/commoncap.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/security/commoncap.c 2007-05-22 16:59:46.000000000 -0700
> @@ -306,7 +306,7 @@ void cap_bprm_apply_creds (struct linux_
>  /* For init, we want to retain the capabilities set
>   * in the init_task struct. Thus we skip the usual
>   * capability rules */
> - if (!is_init(current)) {
> + if (!is_global_init(current)) {

```

At some point we may want to make this more baroque to allow a container init to keep all capabilities relative to it's own container. But we're not there yet.

So actually this might be a good reason to finally repost the user namespace patchset. Ironically I was going to do it during a lull in containers@ traffic, and I'm doing it at perhaps the highest traffic time ever, but it would be good to discuss the semantics of having capabilities "to a namespace" and storing those to a namespace other than your own in your keyring.

-serge

```

> current->cap_permitted = new_permitted;
> current->cap_effective =
>   cap_intersect (new_permitted, bprm->cap_effective);
> Index: lx26-21-mm2/arch/avr32/kernel/traps.c
> =====
> --- lx26-21-mm2.orig/arch/avr32/kernel/traps.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/avr32/kernel/traps.c 2007-05-22 16:59:46.000000000 -0700
> @@ -88,7 +88,7 @@ void _exception(long signr, struct pt_re
>   * generate the same exception over and over again and we get
>   * nowhere. Better to kill it and let the kernel panic.
>   */
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   __sighandler_t handler;
>
>   spin_lock_irq(&current->sigband->siglock);

```

```

> Index: lx26-21-mm2/arch/avr32/mm/fault.c
> =====
> --- lx26-21-mm2.orig/arch/avr32/mm/fault.c 2007-05-22 16:58:38.000000000 -0700
> +++ lx26-21-mm2/arch/avr32/mm/fault.c 2007-05-22 16:59:46.000000000 -0700
> @@ -173,7 +173,7 @@ bad_area:
> if (exception_trace)
>     printk("%s%s[%d]: segfault at %08lx pc %08lx "
>           "sp %08lx ecr %lu\n",
> -       is_init(tsk) ? KERN_EMERG : KERN_INFO,
> +       is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
>         tsk->comm, tsk->pid, address, regs->pc,
>         regs->sp, ecr);
>     _exception(SIGSEGV, regs, code, address);
> @@ -222,7 +222,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> @@ -244,7 +244,7 @@ do_sigbus:
> if (exception_trace)
>     printk("%s%s[%d]: bus error at %08lx pc %08lx "
>           "sp %08lx ecr %lu\n",
> -       is_init(tsk) ? KERN_EMERG : KERN_INFO,
> +       is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
>         tsk->comm, tsk->pid, address, regs->pc,
>         regs->sp, ecr);
>
> Index: lx26-21-mm2/kernel/pid.c
> =====
> --- lx26-21-mm2.orig/kernel/pid.c 2007-05-22 16:59:34.000000000 -0700
> +++ lx26-21-mm2/kernel/pid.c 2007-05-22 16:59:46.000000000 -0700
> @@ -71,6 +71,27 @@ struct pid_namespace init_pid_ns = {
>     .child_reaper = &init_task
> };
>
> +
> +/**
> + * is_global_init - check if a task structure is init
> + * @tsk: Task structure to be checked.
> + *
> + * Check if a task structure is the first user space task the kernel created.
> + */
> +int is_global_init(struct task_struct *tsk)
> +{

```

```
> + return (task_active_pid_ns(tsk) == &init_pid_ns && tsk->pid == 1);
> +}
> +
> +/*
> + * is_container_init:
> + * check whether in the task is init in it's own pid namespace.
> + */
> +int is_container_init(struct task_struct *tsk)
> +{
> + return tsk->pid == 1;
> +}
> +
> +/*
> + * Note: disable interrupts while the pidmap_lock is held as an
> + * interrupt might come in and do read_lock(&tasklist_lock).
```

This looks good. Thanks, Suka.

-serge

---

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

---