

## printk levels for i386 oops code.

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-11/9642.html>

---

*From:* Dave Jones ([davej\\_at\\_redhat.com](mailto:davej_at_redhat.com))

*Date:* 11/30/05

Date: Wed, 30 Nov 2005 00:03:07 -0500

To: [linux-kernel@vger.kernel.org](mailto:linux-kernel@vger.kernel.org)

Especially useful when users have booted with 'quiet'.  
In the regular 'oops' path, we set the console\_loglevel before we start spewing debug info, but we can call the backtrace code from other places now too, such as the spinlock debugging code.

Signed-off-by: Dave Jones <[davej@redhat.com](mailto:davej@redhat.com)>

```
--- vanilla/arch/i386/kernel/traps.c~ 2005-11-29 23:35:29.000000000 -0500
+++ vanilla/arch/i386/kernel/traps.c 2005-11-29 23:51:15.000000000 -0500
@@ -120,7 +120,7 @@ static inline unsigned long print_context
#ifdef CONFIG_FRAME_POINTER
    while (valid_stack_ptr(tinfo, (void *)ebp)) {
        addr = *(unsigned long *) (ebp + 4);
- printk(" [<%08lx>]", addr);
+ printk(KERN_EMERG " [<%08lx>]", addr);
        print_symbol("%s", addr);
        printk("\n");
        ebp = *(unsigned long *)ebp;
@@ -129,7 +129,7 @@ static inline unsigned long print_context
    while (valid_stack_ptr(tinfo, stack)) {
        addr = *stack++;
        if (__kernel_text_address(addr)) {
- printk(" [<%08lx>]", addr);
+ printk(KERN_EMERG " [<%08lx>]", addr);
            print_symbol(" %s", addr);
            printk("\n");
        }
@@ -161,7 +161,7 @@ void show_trace(struct task_struct *task
    stack = (unsigned long *)context->previous_esp;
    if (!stack)
        break;
- printk(" =====\n");
+ printk(KERN_EMERG " =====\n");
    }
}
```

## Linux-Kernel: printk levels for i386 oops code.

```

@@ -178,14 +178,15 @@ void show_stack(struct task_struct *task
    }

    stack = esp;
+ printk(KERN_EMERG);
    for(i = 0; i < kstack_depth_to_print; i++) {
        if (kstack_end(stack))
            break;
        if (i && ((i % 8) == 0))
- printk("\n ");
+ printk("\n " KERN_EMERG);
        printk("%08lx ", *stack++);
    }
- printk("\nCall Trace:\n");
+ printk("\n" KERN_EMERG "Call Trace:\n");
    show_trace(task, esp);
}

@@ -216,18 +217,18 @@ void show_registers(struct pt_regs *regs
    ss = regs->xss & 0xffff;
}
print_modules();
- printk("CPU: %d\nEIP: %04x:[<%08lx>] %s VLI\nEFLAGS: %08lx"
- " (%s) \n",
+ printk(KERN_EMERG "CPU: %d\nEIP: %04x:[<%08lx>] %s VLI\n"
+ "EFLAGS: %08lx (%s) \n",
        smp_processor_id(), 0xffff & regs->xcs, regs->eip,
        print_tainted(), regs->eflags, system_utsname.release);
- print_symbol("EIP is at %s\n", regs->eip);
- printk("eax: %08lx ebx: %08lx ecx: %08lx edx: %08lx\n",
+ print_symbol(KERN_EMERG "EIP is at %s\n", regs->eip);
+ printk(KERN_EMERG "eax: %08lx ebx: %08lx ecx: %08lx edx: %08lx\n",
        regs->eax, regs->ebx, regs->ecx, regs->edx);
- printk("esi: %08lx edi: %08lx ebp: %08lx esp: %08lx\n",
+ printk(KERN_EMERG "esi: %08lx edi: %08lx ebp: %08lx esp: %08lx\n",
        regs->esi, regs->edi, regs->ebp, esp);
- printk("ds: %04x es: %04x ss: %04x\n",
+ printk(KERN_EMERG "ds: %04x es: %04x ss: %04x\n",
        regs->xds & 0xffff, regs->xes & 0xffff, ss);
- printk("Process %s (pid: %d, threadinfo=%p task=%p)",
+ printk(KERN_EMERG "Process %s (pid: %d, threadinfo=%p task=%p)",
        current->comm, current->pid, current_thread_info(), current);
/*
 * When in-kernel, we also print out the stack and code at the
@@ -236,17 +237,17 @@ void show_registers(struct pt_regs *regs
    if (in_kernel) {
        u8 __user *eip;

- printk("\nStack: ");
+ printk("\n" KERN_EMERG "Stack: ");
        show_stack(NULL, (unsigned long*)esp);
    }
}

```

## Linux–Kernel: printk levels for i386 oops code.

```

– printk("Code: ");
+ printk(KERN_EMERG "Code: ");

    eip = (u8 __user *)regs->eip - 43;
    for (i = 0; i < 64; i++, eip++) {
        unsigned char c;

        if (eip < (u8 __user *)PAGE_OFFSET || __get_user(c, eip)) {
– printk(" Bad EIP value.");
+ printk(KERN_EMERG " Bad EIP value.");
            break;
        }
        if (eip == (u8 __user *)regs->eip)
@@ -280,15 +281,15 @@ static void handle_BUG(struct pt_regs *r
    (unsigned long)file < PAGE_OFFSET || __get_user(c, file))
    file = "<bad filename>";

– printk("-----[ cut here ]-----\n");
– printk(KERN_ALERT "kernel BUG at %s:%d!\n", file, line);
+ printk(KERN_EMERG "-----[ cut here ]-----\n");
+ printk(KERN_EMERG "kernel BUG at %s:%d!\n", file, line);

no_bug:
    return;

    /* Here we know it was a BUG but file–n–line is unavailable */
bug:
– printk("Kernel BUG\n");
+ printk(KERN_EMERG "Kernel BUG\n");
}

/* This is gone through when something in the kernel
@@ -318,16 +319,20 @@ void die(const char * str, struct pt_reg
    if (++die.lock_owner_depth < 3) {
        int nl = 0;
        handle_BUG(regs);
– printk(KERN_ALERT "%s: %04lx [##d]\n", str, err & 0xffff, ++die_counter);
+ printk(KERN_EMERG "%s: %04lx [##d]\n", str, err & 0xffff, ++die_counter);
#ifdef CONFIG_PREEMPT
– printk("PREEMPT ");
+ printk(KERN_EMERG "PREEMPT ");
        nl = 1;
#endif
#ifdef CONFIG_SMP
+ if (!nl)
+ printk(KERN_EMERG);
        printk("SMP ");
        nl = 1;
#endif
#ifdef CONFIG_DEBUG_PAGEALLOC

```

## Linux–Kernel: printk levels for i386 oops code.

```

+ if (!nl)
+ printk(KERN_EMERG);
    printk("DEBUG_PAGEALLOC");
    nl = 1;
#endif
@@ -336,7 +341,7 @@ void die(const char * str, struct pt_reg
    notify_die(DIE_OOPS, (char *)str, regs, err, 255, SIGSEGV);
    show_registers(regs);
    } else
- printk(KERN_ERR "Recursive die() failure, output suppressed\n");
+ printk(KERN_EMERG "Recursive die() failure, output suppressed\n");

    bust_spinlocks(0);
    die.lock_owner = -1;
@@ -523,8 +528,8 @@ gp_in_kernel:

static void mem_parity_error(unsigned char reason, struct pt_regs * regs)
{
- printk("Uhhuh. NMI received. Dazed and confused, but trying to continue\n");
- printk("You probably have a hardware problem with your RAM chips\n");
+ printk(KERN_EMERG "Uhhuh. NMI received. Dazed and confused, but trying to continue\n");
+ printk(KERN_EMERG "You probably have a hardware problem with your RAM chips\n");

    /* Clear and disable the memory parity error line. */
    clear_mem_error(reason);
@@ -534,7 +539,7 @@ static void io_check_error(unsigned char
{
    unsigned long i;

- printk("NMI: IOCK error (debug interrupt?)\n");
+ printk(KERN_EMERG "NMI: IOCK error (debug interrupt?)\n");
    show_registers(regs);

    /* Re-enable the IOCK line, wait for a few seconds */
@@ -556,10 +561,10 @@ static void unknown_nmi_error(unsigned c
    return;
}
#endif
- printk("Uhhuh. NMI received for unknown reason %02x on CPU %d.\n",
+ printk(KERN_EMERG "Uhhuh. NMI received for unknown reason %02x on CPU %d.\n",
    reason, smp_processor_id());
- printk("Dazed and confused, but trying to continue\n");
- printk("Do you have a strange power saving mode enabled?\n");
+ printk(KERN_EMERG "Dazed and confused, but trying to continue\n");
+ printk(KERN_EMERG "Do you have a strange power saving mode enabled?\n");
}

static DEFINE_SPINLOCK(nmi_print_lock);
@@ -576,11 +581,11 @@ void die_nmi (struct pt_regs *regs, cons
    * to get a message out.
    */

```

## Linux-Kernel: printk levels for i386 oops code.

```
    bust_spinlocks(1);
- printk(msg);
+ printk(KERN_EMERG "%s", msg);
    printk(" on CPU%d, eip %08lx, registers:\n",
          smp_processor_id(), regs->eip);
    show_registers(regs);
- printk("console shuts up ...\n");
+ printk(KERN_EMERG "console shuts up ...\n");
    console_silent();
    spin_unlock(&nmi_print_lock);
    bust_spinlocks(0);
@@ -993,8 +998,8 @@ asmlinkage void math_state_restore(struc

asmlinkage void math_emulate(long arg)
{
- printk("math-emulation not enabled and no coprocessor found.\n");
- printk("killing %s.\n",current->comm);
+ printk(KERN_EMERG "math-emulation not enabled and no coprocessor found.\n");
+ printk(KERN_EMERG "killing %s.\n",current->comm);
    force_sig(SIGFPE,current);
    schedule();
}
-
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>