

Re: Serial related oops

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg06796.html>

- *From:* Jose Goncalves <jose.goncalves@xxxxxxx>
 - *Date:* Mon, 19 Feb 2007 17:54:52 +0000
-

Russell King wrote:

On Mon, Feb 19, 2007 at 04:29:39PM +0000, Jose Goncalves wrote:

Russell King wrote:

On Tue, Feb 20, 2007 at 02:48:14PM +0000, Frederik Deweerdt wrote:

(trimmed tie-fei.zang from the CC, added by mistake)

On Mon, Feb 19, 2007 at 02:35:20PM +0000, Russell King wrote:

Neither did I, but introducing printk's through the function, we narrowed the problem to this part of the code. And removing it makes the problem go away. We inserted 37 printk's in the function

Re: Serial related oops

body, and
Jose
bisected
those until
the problem
went away.

Well, there's still little clue
about why this is causing a
NULL pointer
dereference. The only thing
I can think is that somehow
performing
this test is causing a power
glitch to your CPU, causing
its registers
to get corrupted, and which
results in it doing a NULL
pointer deref.

That may be the case, indeed.

But if the problem was a power glitch I should get Oops with or without
printk() inserted, shouldn't I?

That depends if the printk() changes the timing such that it doesn't occur.
Don't know, I'm only grasping at straws due to the lack of any concrete
information.

If you see other tests to be performed...

Maybe adding some delays in that bit of code? I'm sure
you've already
thought of that though. Since no one has a proper
understanding of the
problem, the only suggestions possible are mere shots in the
dark.

Re: Serial related oops

I'm no kernel expert, but it's not possible to trace what is the instruction that is causing the NULL pointer dereference?

The reported dump shows that the kernel tried to access virtual address 0, and the instruction pointer seems to be the cause of that – it has a value of zero in that dump.

The call trace indicates that the last function was called from around "uart_startup+0x63/0xf4" which is probably the indirect function call to serial8250_startup(). That's unconfirmed – the only way to get it confirmed is if you could dump the entire uart_startup() function.

```
$ grep uart_startup System.map
(address) T uart_startup
$ objdump -r -d vmlinux --start-addr=0x<address> --stop-addr=0x<address+256>
```

The grep should get you the address of uart_startup. Replace <address> with that value and <address+256> with the value plus 256 (0x100) and mail the result.

Result is attached.

José Gonçalves

vmlinux-2.6.16.38-mtm4-debug: file format elf32-i386

Disassembly of section .text:

```
c01ba437 <uart_startup>:
c01ba437: 55 push %ebp
c01ba438: 57 push %edi
c01ba439: 56 push %esi
c01ba43a: 53 push %ebx
c01ba43b: 8b 7c 24 14 mov 0x14(%esp),%edi
c01ba43f: 31 d2 xor %edx,%edx
c01ba441: 8b 5f 10 mov 0x10(%edi),%ebx
c01ba444: 8b 77 14 mov 0x14(%edi),%esi
c01ba447: 83 7b 10 00 cmpl $0x0,0x10(%ebx)
c01ba44b: 0f 88 d3 00 00 00 js c01ba524 <uart_startup+0xed>
c01ba451: 8b 03 mov (%ebx),%eax
c01ba453: 0f ba a8 b4 00 00 00 btsl $0x1,0xb4(%eax)
c01ba45a: 01
c01ba45b: 83 7e 60 00 cmpl $0x0,0x60(%esi)
c01ba45f: 0f 84 bf 00 00 00 je c01ba524 <uart_startup+0xed>
c01ba465: 83 7b 04 00 cmpl $0x0,0x4(%ebx)
c01ba469: 75 28 jne c01ba493 <uart_startup+0x5c>
```

Re: Serial related oops

Re: Serial related oops

```
c01ba46b: b8 d0 00 00 00 mov $0xd0,%eax
c01ba470: e8 36 c7 f6 ff call c0126bab <get_zeroed_page>
c01ba475: ba f4 ff ff ff mov $0xffffffff4,%edx
c01ba47a: 85 c0 test %eax,%eax
c01ba47c: 0f 84 a2 00 00 00 je c01ba524 <uart_startup+0xed>
c01ba482: 89 43 04 mov %eax,0x4(%ebx)
c01ba485: c7 43 0c 00 00 00 00 movl $0x0,0xc(%ebx)
c01ba48c: c7 43 08 00 00 00 00 00 movl $0x0,0x8(%ebx)
c01ba493: 8b 46 64 mov 0x64(%esi),%eax
c01ba496: 56 push %esi
c01ba497: ff 50 24 call *0x24(%eax)
c01ba49a: 89 c5 mov %eax,%ebp
c01ba49c: 58 pop %eax
c01ba49d: 85 ed test %ebp,%ebp
c01ba49f: 75 6d jne c01ba50e <uart_startup+0xd7>
c01ba4a1: 83 7c 24 18 00 cmpl $0x0,0x18(%esp)
c01ba4a6: 74 36 je c01ba4de <uart_startup+0xa7>
c01ba4a8: 6a 00 push $0x0
c01ba4aa: 57 push %edi
c01ba4ab: e8 5f 02 00 00 call c01ba70f <uart_change_speed>
c01ba4b0: 8b 03 mov (%ebx),%eax
c01ba4b2: 8b 40 64 mov 0x64(%eax),%eax
c01ba4b5: 59 pop %ecx
c01ba4b6: 5f pop %edi
c01ba4b7: f7 40 08 0f 10 00 00 testl $0x100f,0x8(%eax)
c01ba4be: 74 1e je c01ba4de <uart_startup+0xa7>
c01ba4c0: 9c pushf
c01ba4c1: 5f pop %edi
c01ba4c2: fa cli
c01ba4c3: 8b 46 58 mov 0x58(%esi),%eax
c01ba4c6: 89 c2 mov %eax,%edx
c01ba4c8: 83 ca 06 or $0x6,%edx
c01ba4cb: 89 56 58 mov %edx,0x58(%esi)
c01ba4ce: 39 d0 cmp %edx,%eax
c01ba4d0: 74 0a je c01ba4dc <uart_startup+0xa5>
c01ba4d2: 8b 46 64 mov 0x64(%esi),%eax
c01ba4d5: 52 push %edx
c01ba4d6: 56 push %esi
c01ba4d7: ff 50 04 call *0x4(%eax)
c01ba4da: 58 pop %eax
c01ba4db: 5a pop %edx
c01ba4dc: 57 push %edi
c01ba4dd: 9d popf
c01ba4de: f6 43 13 04 testb $0x4,0x13(%ebx)
c01ba4e2: 74 17 je c01ba4fb <uart_startup+0xc4>
c01ba4e4: fa cli
c01ba4e5: 8b 46 64 mov 0x64(%esi),%eax
c01ba4e8: 56 push %esi
c01ba4e9: ff 50 08 call *0x8(%eax)
c01ba4ec: 5e pop %esi
c01ba4ed: a8 20 test $0x20,%al
```

Re: Serial related oops

Re: Serial related oops

```
c01ba4ef: 75 09 jne c01ba4fa <uart_startup+0xc3>
c01ba4f1: 8b 03 mov (%ebx),%eax
c01ba4f3: 80 88 c4 00 00 00 02 orb $0x2,0xc4(%eax)
c01ba4fa: fb sti
c01ba4fb: 81 4b 10 00 00 00 80 orl $0x80000000,0x10(%ebx)
c01ba502: 8b 03 mov (%ebx),%eax
c01ba504: 0f ba b0 b4 00 00 00 btrl $0x1,0xb4(%eax)
c01ba50b: 01
c01ba50c: eb 14 jmp c01ba522 <uart_startup+0xeb>
c01ba50e: 6a 15 push $0x15
c01ba510: e8 b0 c3 f5 ff call c01168c5 <capable>
c01ba515: 85 c0 test %eax,%eax
c01ba517: 59 pop %ecx
c01ba518: 0f 94 c0 sete %al
c01ba51b: 0f b6 c0 movzbl %al,%eax
c01ba51e: f7 d8 neg %eax
c01ba520: 21 c5 and %eax,%ebp
c01ba522: 89 ea mov %ebp,%edx
c01ba524: 89 d0 mov %edx,%eax
c01ba526: 5b pop %ebx
c01ba527: 5e pop %esi
c01ba528: 5f pop %edi
c01ba529: 5d pop %ebp
c01ba52a: c3 ret
```

```
c01ba52b <uart_shutdown>:
c01ba52b: 57 push %edi
c01ba52c: 56 push %esi
c01ba52d: 53 push %ebx
c01ba52e: 8b 44 24 10 mov 0x10(%esp),%eax
c01ba532: 8b 58 10 mov 0x10(%eax),%ebx
c01ba535: 8b 70 14 mov 0x14(%eax),%esi
Disassembly of section .init.text:
Disassembly of section .altinstr_replacement:
Disassembly of section .exit.text:
```