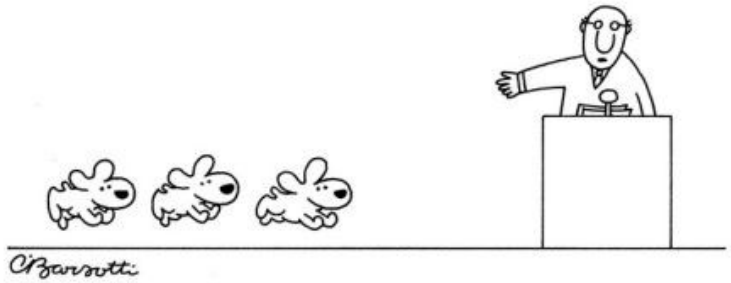
Debugging

Oops - what happened?

ECE 373

But first...

- `write()` in kernel takes `char __user *buf`
  - How to convert?
- How to convert in normal C programs?
  - `atoi()`
  - Manpage!!
- How to find?
  - Google is your friend
  - LXR is a better friend
  - https://elixir.bootlin.com/linux/latest/source/include/linux/kernel.h#L203



"But first, a distraction."

What does a bug look like?

- Kernel panic – the machine is dead
- Odd messages on the console or in the `/var/log/{messages|syslog}` file
- The network messages are garbled
- The light won't stop blinking
- The robot fell over

Stack Trace

- Printed to console when something bad happens

```
Fedora release 14 (Laughlin)
Kernel 2.6.35.13-92.fc14.x86_64 on an x86_64 (/dev/ttyS0)

ppwaskie-fed14-vm login: [  585.128074] hello kernel...
[  585.129248] BUG: unable to handle kernel NULL pointer
dereference at (null)
[  585.130106] IP: [<ffffffffa003a01b>]
ece_foobar_init+0x1b/0x2f [ece_foobar]
[  585.130106] PGD 37c81067 PUD 37f3d067 PMD 0
[  585.130106] Oops: 0002 [#1] SMP
[  585.130106] last sysfs file:
/sys/devices/pci0000:00/0000:00:01.2/usb1/1-1/dm
[  585.130106] CPU 0
[  585.130106] Modules linked in: ece_foobar(+) tcp_lp fuse
sunrpc ip6t_REJECT ]
```

Stack Trace, newer

```
[  378.411171] RIP: 0033:0x7f22facae89d
[  378.411537] Code: 00 c3 66 2e 0f 1f 84 00 00 00 00 00 90 f3 0f 1e fa 48 89 f8 48 89 f7 48 89 d6 4
8 89 ca 4d 89 c2 4d 89 c8 4c 8b 4c 24 08 0f 05 <48> 3d 01 f0 ff ff 73 01 c3 48 8b 0d c3 f5 0c 00 f7
d8 64 89 01 48
[  378.412912] RSP: 002b:00007ffc25dfdfa8 EFLAGS: 00000246 ORIG_RAX: 0000000000000139
[  378.413288] RAX: ffffffffffffffda RBX: 000055d01b6427a0 RCX: 00007f22facae89d
[  378.413780] RDX: 0000000000000000 RSI: 000055d01996f358 RDI: 0000000000000003
[  378.414217] RBP: 0000000000000000 R08: 0000000000000000 R09: 00007f22fad82260
[  378.414694] R10: 0000000000000003 R11: 0000000000000246 R12: 000055d01996f358
[  378.415075] R13: 0000000000000000 R14: 000055d01b642760 R15: 0000000000000000
[  378.415447] Modules linked in: ece373_foobar(OE+) vboxvideo(OE) nls_iso8859_1 snd_intel8x0 snd_ac
97_codec ac97_bus snd_pcm snd_seq_midi snd_seq_midi_event snd_rawmidi snd_seq intel_rapl_msr intel_r
apl_common crct10dif_pclmul ghash_clmulni_intel snd_seq_device snd_timer joydev aesni_intel snd cryp
to_simd cryptd glue_helper rapl input_leds serio_raw soundcore vboxguest(OE) mac_hid sch_fq_codel vm
wgfx ttm drm_kms_helper cec rc_core fb_sys_fops syscopyarea sysfillrect sysimgblt parport_pc ppdev l
p parport drm ip_tables x_tables autofs4 hid_generic usbhid hid crc32_pclmul psmouse ahci libahci iz
c_piix4 e1000 pata_acpi video [last unloaded: ece373_foobar]
[  378.418857] CR2: 0000000000000008
[  378.419603] ---[ end trace 02aeebd0dcbf5b3b ]---
[  378.420144] RIP: 0010:ece373_foobar_init+0x15/0x1000 [ece373_foobar]
[  378.420557] Code: Unable to access opcode bytes at RIP 0xffffffffc0992feb.
[  378.420888] RSP: 0018:ffffa66dc2d5bc60 EFLAGS: 00010246
[  378.421187] RAX: 0000000000000012 RBX: 0000000000000000 RCX: 0000000000000000
[  378.421626] RDX: 0000000000000000 RSI: ffff925afdc18cd0 RDI: ffff925afdc18cd0
[  378.421942] RBP: ffffa66dc2d5bc60 R08: ffff925afdc18cd0 R09: 0000000000000004
[  378.422255] R10: 0000000000000000 R11: 0000000000000001 R12: ffffffffc0993000
[  378.422585] R13: ffff925abe5b2d40 R14: ffffa66dc2d5be70 R15: 0000000000000000
[  378.422978] FS:  00007f22fab69540(0000) GS:ffff925afdc00000(0000) knlGS:0000000000000000
[  378.423505] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[  378.423845] CR2: ffffffffc0992feb CR3: 0000000011c48004 CR4: 00000000000606f0
Killed
ojw@ece373-ubuntu:~/ece373/examples$
```

More indicators...

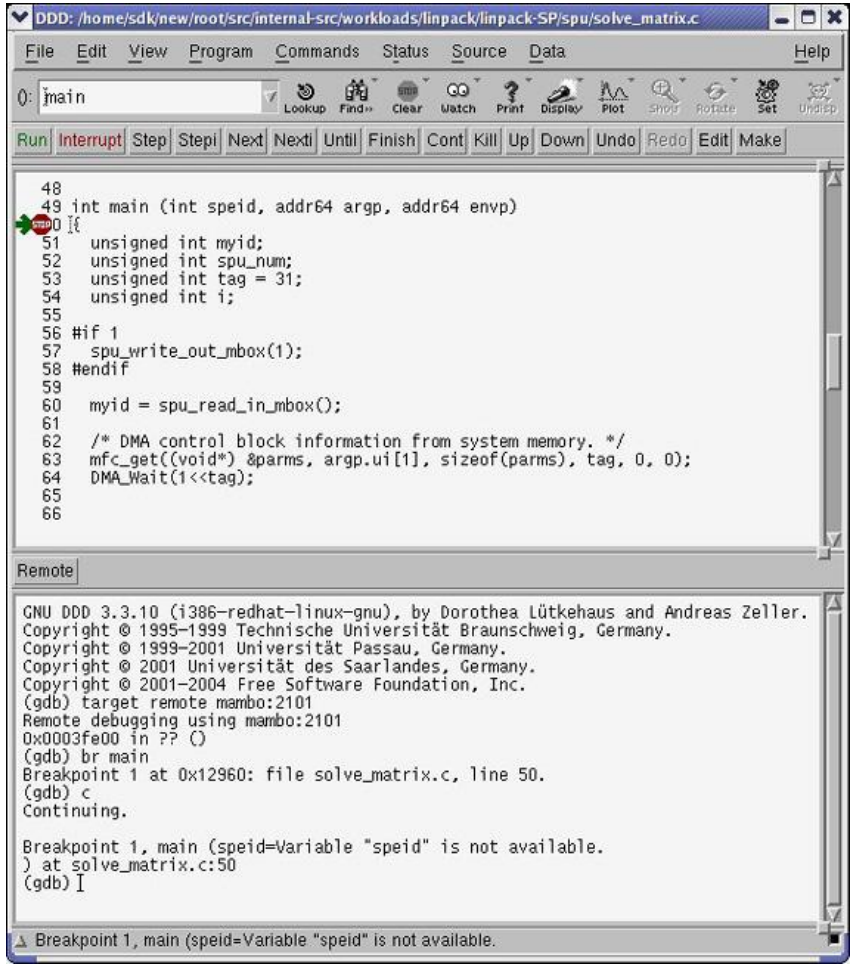
- "Uh, that was weird..."
- "How did that happen?"
- "Why did it do that?"
- “Doesn’t do that on my machine…”

- Now what?

kgdb - Kernel source debugger

- Support in kernel
- DDD user interface
- Remote debugging

- Can be hard to set up
- Need know where to start looking



Gathering Clues

- What are the symptoms?
- How do you reproduce the problem
  - Easy, 100% reproducible?
  - Only happens once in a blue moon?
  - Special HW or SW involved?
- What SW versions?
- What else is going on in the system?

printk()

- Easy to use
  - Sprinkle around code while debugging
  - Print interesting information
    - current values of interesting variables
    - on entry/exit of interesting routines
  - Recompile/relink/test is fast now-a-days
- Don't forget to remove when done
  - Linux community frowns on noisy drivers

printk()

- `printk(KERN_INFO "chainlink=%d\n", chain);`
  - `KERN_EMERG, KERN_ALERT, KERN_CRIT, KERN_ERR, KERN_WARNING, KERN_NOTICE, KERN_INFO, KERN_DEBUG`
- `tail -f /var/log/messages`
  - filtered by kernel param "`loglevel=n`"
    - See `<linuxsrc>/Documentation/kernel-parameters.txt`
  - saved on disk
- `dmesg`
  - not filtered, all msgs show up
  - not saved on disk

`printk()` takes time

- Buffered data not saved before crash
- Print slows time-sensitive operations
  - Use "global" status variables, counters, print later
- Print too much on loops
  - Print only every 100th time

`pr_info()` and friends

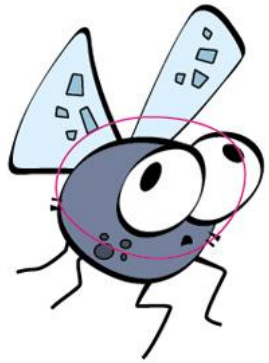
- Friendly wrappers around printk
- Annotates and stamps who printed the message
- Can be compiled out based on debug levels
- Makes printk more portable

`trace_printk()`

- Part of the ftrace function-tracer framework
- Unbuffered, has much less impact to performance/timing
- More desirable to use during interrupts
- Can be enabled/disabled on the fly

WARN, BUG

- Code warnings
  - `BUG(), BUG_ON(expr)`
  - `WARN(), WARN_ON(expr), WARN_ONCE()`
    - https://elixir.bootlin.com/linux/latest/source/drivers/net/ethernet/intel/ice/ice_base.c#L206
- `BUG()` stops the kernel thread
- Both produce stack dump output



objdump -S -d ece_foobar.o

- Decode exact spot of stack dump cause
- Need to compile with '-g' for debug symbols

```
    printk(KERN_INFO "%s: cmd=%d\n", __FUNCTION__, cmd);
2c: 31 c0                   xor    %eax,%eax
2e: 48 c7 c6 00 00 00 00    mov    $0x0,%rsi
35: 48 c7 c7 00 00 00 00    mov    $0x0,%rdi
3c: 89 da                   mov    %ebx,%edx
3e: e8 00 00 00 00          callq  43 <timer_cb+0x43>
    switch (cmd) {
43: 83 fb 01                cmp    $0x1,%ebx
46: 74 20                   je     68 <timer_cb+0x68>
48: 7e 36                   jle    80 <timer_cb+0x80>
4a: 83 fb 02                cmp    $0x2,%ebx
4d: eb 01                   jmp    50 <timer_cb+0x50>
4f: 90                      nop
50: 74 47                   je     99 <timer_cb+0x99>
52: 83 fb 03                cmp    $0x3,%ebx
55: eb 01                   jmp    58 <timer_cb+0x58>
57: 90                      nop
58: 74 52                   je     ac <timer_cb+0xac>
```

objdump -S -d ece_foobar.o

```
pjw@ece373-ubuntu:~/ece373/examples$ objdump -S -d ece373_foobar.ko

ece373_foobar.ko:     file format elf64-x86-64


Disassembly of section .init.text:

0000000000000000 <init_module>:
   0:   e8 00 00 00 00          callq  5 <init_module+0x5>
   5:   55                      push   %rbp
   6:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   d:   48 89 e5                mov    %rsp,%rbp
  10:   e8 00 00 00 00          callq  15 <init_module+0x15>
  15:   48 8b 34 25 08 00 00    mov    0x8,%rsi
  1c:   00
  1d:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
  24:   e8 00 00 00 00          callq  29 <init_module+0x29>
  29:   31 c0                   xor    %eax,%eax
  2b:   5d                      pop    %rbp
  2c:   c3                      retq

Disassembly of section .exit.text:

0000000000000000 <cleanup_module>:
   0:   55                      push   %rbp
   1:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   8:   48 89 e5                mov    %rsp,%rbp
   b:   e8 00 00 00 00          callq  10 <cleanup_module+0x10>
  10:   5d                      pop    %rbp
  11:   c3                      retq
pjw@ece373-ubuntu:~/ece373/examples$
```

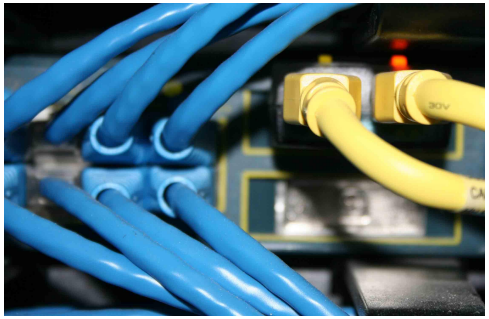
Ethtool

- `ethtool -i`: network device info

```
$ ethtool -i eth0
driver: e1000
version: 7.3.21-k8-NAPI
firmware-version: N/A
bus-info: 0000:02:01.0
```

- `ethtool -S`: network statistics

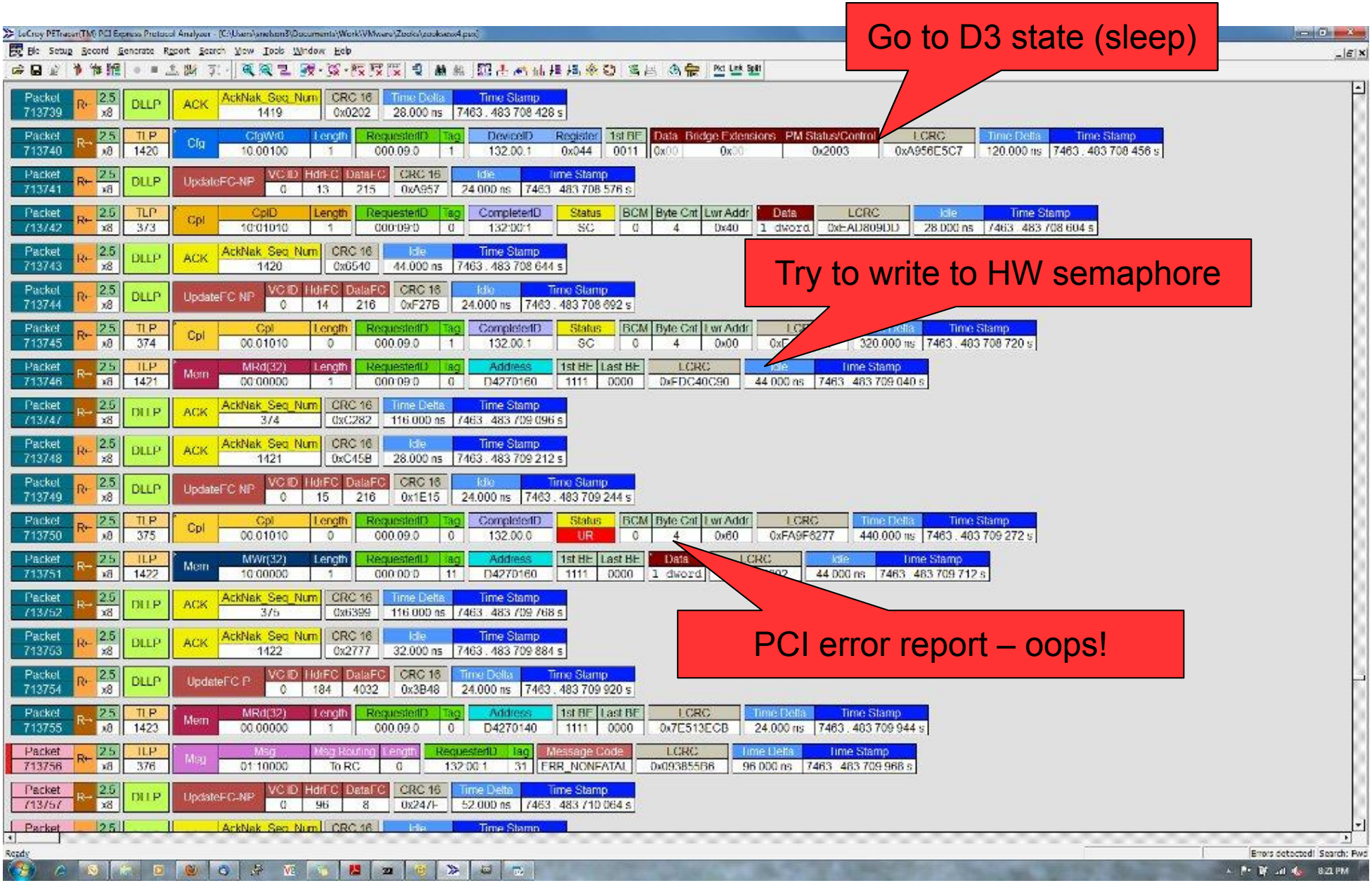
```
$ sudo ethtool -S eth0
NIC statistics:
     rx_packets: 1200
     tx_packets: 648
     rx_bytes: 530648
     tx_bytes: 87288
     rx_broadcast: 0
     tx_broadcast: 0
     rx_multicast: 0
     tx_multicast: 0
     rx_errors: 23
     tx_errors: 0
     tx_dropped: 0
     multicast: 0
     collisions: 0
     rx_length_errors: 47
     rx_over_errors: 0
     rx_crc_errors: 35
```



PCI bus trace

- Hardware to capture PCI data on the bus
  - Case 1: PCI error on initialization
    - Similar network chips, slightly different register sets
    - Code for 82599 was writing to non-existent config registers on 82598, cause "odd" things to happen
  - Case 2: Occasional PCI error on system shutdown
    - Network board put into D3 (sleep) mode
    - Check-status timer expired, tried to read from sleeping board

PCI case 2


Go to D3 state (sleep)
Try to write to HW semaphore
PCI error report – oops!

kdump

- Kernel crash dump capture facility
- Not straight forward to configure
- Requires deep kernel bits to work
- Target scratch device
- Very similar to core dump

crash

- Used to analyze kdump crashes
- Similar to gdb
- Requires environment to get running

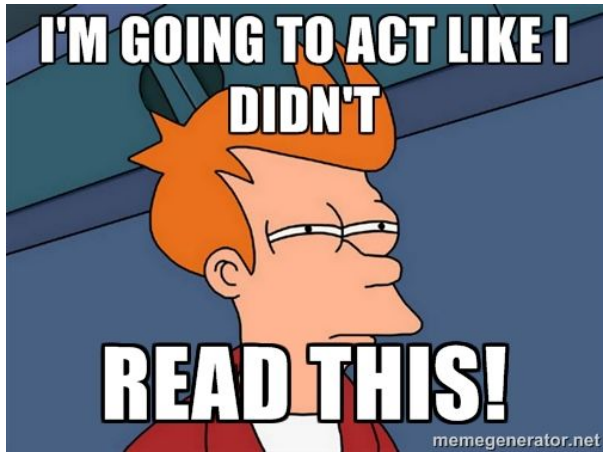
Other

- /proc
  - Interrupts, iomem, ioports,
- watch -d "cmd"
  - Repeats commands, show differences
- Diff from previously working code

Time for a Scooby Snack!

- Lots of tools for sniffing out problems
- Gather data before fixing
- Use repeatable tests
  - First to track the problem...
  - ... then to prove it is fixed
- When stymied
  - take a break, ask for suggestions, read up ...
  - ... and try, try again.

Reading



- Debugging:
  - LDD3, chapter 4
  - ELDD, chapter 21
    - Loose focus on kgdb, kexec, kdump
- Upcoming reading – Memory!:
  - Linux Drivers, Chapters 11 and 12
  - LDD3, Chapter 8
  - ELDD, Pages 49 - 51