ECE 471 – Embedded Systems
Lecture 21

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

28 October 2019

Announcements

- Don't forget SPI homework HW#7
- Update on NASA talk

HW#6 Review – i2c Questions

- Protocol – what is missing?
  - Question is the protocol, not the code. So you didn't implement `read_scl()`, when would that be useful?
  - No code to act as device? (true but that wasn't goal)
  - No support for reads
  - No clock-stretching
  - No Arbitration
  - No way to specify the device address
- Handle all errors?

- Again, no arbitration, etc.
- Also not check results of write functions
- Did not mean can it detect code errors like forcing lines high.

HW#6 Review – Other Questions

- How make code only visible in own file?
  - `static`
  - Why do that?
    optimization (compiler can inline)
    size (compiler can remove unused functions) is it just annoying you with unused functions? how could it ever be useful to know if a func/ variable is unused?
- How does Linux know there's an i2c bus?
  - Does it probe it?

- Can you enumerate it?
- Could the firmware tell you?
- When you compile the kernel do you compile a special Pi kernel that knows a bcm2835 i2c port lives at a certain address?
- What about kernel modules?
- Is the knowledge provided at boot by a “device tree” file: `arch/arm/boot/dts/bcm283x.dtsi`

```
i2c0: i2c@7e205000 {
compatible = "brcm,bcm2835-i2c";
reg = <0x7e205000 0x200>;
interrupts = <2 21>;
clocks = <&clocks BCM2835_CLOCK_VPU>;
#address-cells = <1>;
#size-cells = <0>;
status = "disabled";
};
```

HW#6 Review – Linux Fun

- Interrupt sources: VCHIQ doorbell
- Yes command – mostly to answer things like fsck that ask a lot of obvious questions.
  Load testing, maybe, but that wasn't really the original design.

HW#7 Followup

SPI Sample Code Review

```
#define LENGTH 3
int result;
struct spi_ioc_transfer spi;
unsigned char data_out[LENGTH]={0x1,0x2,0x3};
unsigned char data_in[LENGTH];

/* kernel doesn't like it if stray values, even in padding */
memset(&spi,0,sizeof(struct spi_ioc_transfer));

/* Setup full-duplex transfer of 3 bytes */
spi.tx_buf   = (unsigned long)&data_out;
spi.rx_buf   = (unsigned long)&data_in;
spi.len      = LENGTH;
spi.delay_usecs      = 0 ;
spi.speed_hz         = 100000 ;
spi.bits_per_word    = 8 ; spi.cs_change        = 0 ;

/* Run one full-duplex transaction */
result = ioctl(spi_fd, SPI_IOC_MESSAGE(1), &spi) ;
```

Zeroed Structs in Kernel ABI

- Why is the kernel erroring out if the empty “pad” bit not zero?
  - Forward compatibility. You want to make sure that any empty bits stay that way.
  - If you want to add new functionality in the future you have to ensure reserved bits are all zero, otherwise old programs will do unexpected things (or break) if they had been accidentally setting those bits.

- So why were the pad bits non-zero? Bad luck. Local struct allocated on the stack, so if there were old values on the stack the pad value could be non-zero.

Floating Point in C

- Converting int to floating point:

```
int value=45;
double temp;

temp=value;      // works
temp=(float)value;  // casts make the conversion explicit
            // but can potentially hide bugs
```

- `float` vs `double`

  float is 32-bit, double 64-bit

- Constants 9/5 vs 9.0/5.0

The first is an integer so just “1”. Second is expected 1.8.

- Printing. First prints a double. Second prints a double with only 2 digits after decimal.

```
printf("%lf\n",temp);
printf("%.2lf\n",temp);
```

Is Regular Linux a RTOS

- Not really
- Can do priorities ("nice") but the default ones are not RT.

Real Time Linux

- Project to have a small supervisor RTOS and run Linux as a process
- Code that needed a compatible OS interface could call into this process-Linux, but it could always be preempted
- Not supported anymore?

PREEMPT Kernel

- Linux PREEMPT_RT
- Faster response times
- Remove all unbounded latencies
- Change locks and interrupt threads to be pre-emptible
- Have been gradually merging changes upstream

Typical kernel, when can you pre-empt

- When user code running
- When a system call or interrupt happens
- When kernel code blocks on mutex (lock) or voluntarily yields
- If a high priority task wants to run, and the kernel is running, it might be hundreds of milliseconds before you get to run

- Pre-empt patch makes it so almost any part of kernel can be stopped (pre-empted). Also moves interrupt routines into pre-emptible kernel threads.

Linux PREEMPT Kernel

- What latencies can you get?
  10-30us on some x86 machines
- Depends on firmware; SMI interrupts (secret system mode, can't be blocked, emulate USB, etc.)
  Slow hardware; CPU frequency scaling; nohz
- Special patches, recompile kernel
- Priorities
  - Linux Nice: -20 to 19 (lowest), use nice command
  - Real Time: 0 to 99 (highest)

- Appears in ps as 0 to 139?

Changes to your code

- What do you do about unknown memory latency?
  - mlockall() memory in, start threads and touch at beginning, avoid all causes of pagefaults.
- What do you do about priority?
  - Use POSIX interfaces, no real changes needed in code, just set higher priority
  - See the `chrt` tool to set priorities.
- What do you do about interrupts?
  - See next

Interrupts

- Why are interrupts slow?
- Shared lines, have to run all handlers
- When can they not be pre-empted? IRQ disabled? If a driver really wanted to pause 1ms for hardware to be ready, would often turn off IRQ and spin rather than sleep
- Higher priority IRQs? FIR on ARM?
- Top Halves / Bottom Halves
- Unrelated, but hi-res timers

Co-operative real-time Linux

- Xenomai
- Linux run as side process, sort of like hypervisor

Non-Linux RTOSes

- Interesting reference: `https://rtos.com/rtos/`
- Often are much simpler than Linux
- Some only need a few kilobytes of overhead
- Really, just replacements for an open-coded main loop that did a few tasks sequentially. (Effectively round-robin). Can possibly get better response if you multitask.
- Provide fast context-switching, interrupt handling,

process priority (scheduling), and various locking/mutex libraries

List of some RTOSes

- Vxworks
- Neutrino
- Free RTOS
- Windows CE
- MongooseOS (recent LWN article?)
- ThreadX (in the Pi GPU)