ECE 471 – Embedded Systems
Lecture 21

Vince Weaver

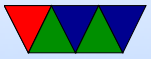
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

23 October 2017

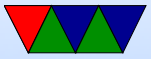
Announcements

- Project coming
- Don't forget SPI homework HW#7



HW#6 Review

- Code all worked for the most part
  You'll note the bitbang code is a lot slower, mostly because we use usleep to do the delays and also the slow GPIO interface. i2c is forgiving
- Compiler warnings – mostly due to static keyword
  - Static on variable means global variable (keeps its value) but local scope (only visible inside of function)
  - Static on function means function is of local scope (only visible in file)

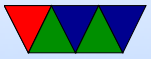


symbol not exported

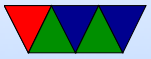
warning if you stop using the function

can be inlined (for speed)

- Questions:
  - All the protocols? No reads. No clock-stretching. No arbitration. No way to change address.
  - Handle all errors? Again, no arbitration, etc.
  - Brakes – hard real time?
  - Tuner – soft real time
  - Video – firm real time
  - Interrupts. Doorbell.

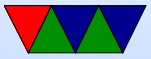


- Yes command – mostly to answer things like fsck that ask a lot of obvious questions.
  Load testing, maybe, but that wasn't really the original design.



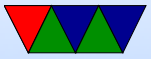
Project Preview

- Can work in groups
- Embedded system (any type, not just Pi)
- Written in any language (asm, C, python, C++, Java, etc.)
- Do some manner of input and some manner of output using the various capabilities we discussed
- I have a large amount of i2c, spi, and other devices that



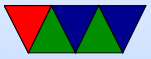
you can borrow if you want to try anything interesting.

- Past projects: games, robots, weather stations, motor controllers, music visualization, etc.
- Will be a final writeup, and then a 10 minute presentation and demo in front of the class during last week of classes.



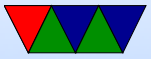
PREEMPT Kernel

- Linux PREEMPT_RT
- Faster response times
- Remove all unbounded latencies
- Change locks and interrupt threads to be pre-emptible

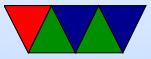


Typical kernel, when can you pre-empt

- 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

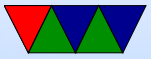


- Pre-empt patch makes it so almost any part of kernel can be stopped (pre-empted). Also moves interrupt routines into pre-emptible kernel threads.

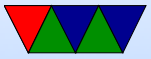


Linux PREEMPT Kernel

- 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 and like)' 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)

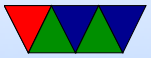


- Appears in ps as 0 to 139?



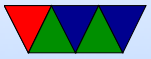
Changes to your code

- 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



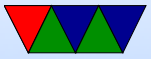
Interrupts

- 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



Other RTOSes

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