ECE 471 – Embedded Systems Lecture 22

Vince Weaver

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

vincent.weaver@maine.edu

26 October 2022

Announcements

- Don't forget HW#6
- Keep thinking about projects, topic due next Friday.
- HW#5 was finally graded
  Watch those compiler warnings, also be sure to comment code

Memory Allocation in Embedded Systems

Memory Allocation – Dynamic

- Using `malloc()/calloc()` or `new()`
- In C have to make sure you `free()` at end
- Downsides:
  - What to do if fails?
    Can you handle that? What if error code also tries to alloc?
  - Timing overhead? Is it deterministic?
    Especially problem with high-level languages and garbage collection

- ◦ Fragmentation: when there's plenty of RAM free but it's in small chunks when you need a large chunk

Memory Allocation – Static

- Allocate all memory you need at startup
- Fail early
- This isn't always possible, but avoids issues with failure, overhead, etc.

Is Regular Linux a RTOS

- Not really
- Can do priorities ("nice") but the default ones are not RT.
- Aside, "nice" comes from old UNIX multi-user days, when you could be nice and give your long-running jobs a low-priority so they wouldn't interfere with other people doing interactive tasks

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?

Linux code that's RT Friendly

- What do you do about unknown memory latency?
  - mlockall() memory in, start threads and touch at beginning, avoid all causes of pagefaults (so no millisecond delays if memory swapped to disk)
- 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

Next up is SPI

Start early on it as there's more than one lecture of material