ECE 471 – Embedded Systems
Lecture 22

Vince Weaver

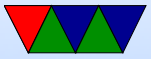
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

27 October 2021

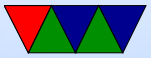
Announcements

- Don't forget SPI homework HW#7
- Don't forget project topics (due next Friday, the 5th)



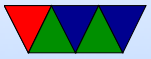
Scheduler example

- Simple: In order the jobs arrive
- Static: (RMS) Rate Monotonic Scheduling – shortest first
- Dynamic: (EDF) Earliest deadline first
- Three tasks come in
  - A: finish in 10s, 4s long
  - B: finish in 3s, 2s long
  - C: finish in 5s, 1s long
- Can they meet the deadline?



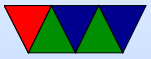
- There is a large body of work on scheduling algorithms.

In-order	A	A	A	A	B	B	C	-	-	-
RMS	C	B	B	A	A	A	A	-	-	-
EDF	B	B	C	A	A	A	A	-	-	-



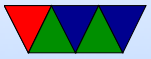
Locking

- When shared hardware/software and more than one thing might access at once
- Multicore: thread 1 read temperature, write to temperature variable
  thread 2 read temperature variable to write to display
  let's say it's writing 3 digit ASCII. Goes from 79 to 80. Will you always get 79 or 80? Can you get 70 or 89?
- How do you protect this? With a lock. Special data structure, allows only one access to piece of memory,



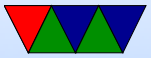
others have to wait.

- Can this happen on single core? Yes, what about interrupts.
- Implemented with special instructions, in assembly language
- Usually you will use a library, like pthreads
- mutex/spinlock
- Atomicity



Priority Inversion Example

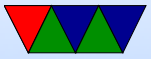
- Task priority 3 takes lock on some piece of hardware (camera for picture)
- Task 2 fires up and pre-empts task 3
- Task 1 fires up and pre-empts task 1, but it needs same HW as task 3. Waits for it. It will never get free. (camera for navigation?)
- Space probes have had issues due to this.



Real Time without an O/S

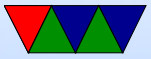
Often an event loop. All parts have to be written so deadlines can be met.

```
main() {
    while(1) {
        do_task1(); // read sensor
        do_task2(); // react to sensor
        do_task3(); // update GUI (low priority)
    }
}
```



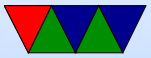
Bare Metal

- What if want priorities?
- Have GUI always run, have the other things happen in timer interrupt handler?
- What if you have multiple hardware all trying to use interrupts (network, serial port, etc)
- At some point it's easier to let an OS handle the hard stuff



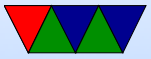
Real Time Operating System

- Can provide multi-tasking/context-switching
- Can provide priority-based scheduling
- Can provide low-overhead interrupts
- Can provide locking primitives



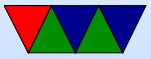
Hard Real Time Operating System

- Can it be hard real time?
- Is it just some switch you can throw? (No)
- Simple ones can be mathematically provable
- Otherwise, it's a best effort



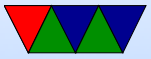
Priority Based, like Vxworks

- Each task has priority 0 (high) to 255 (low)
- When task launched, highest priority gets to run
- Other tasks only get to run when higher is finished or yields
- What if multiple of same priority? Then go round-robin or similar



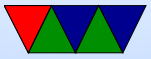
Free RTOS

- Footprint as low as 9K
- Pre-emptive or co-op multitasking
- Task priority
- Semaphores/Mutexes
- Timers
- Stack overflow protection
- Inter-process communication (queues, etc)
- Power management support
- Interrupts (interrupt priority)



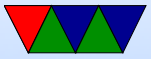
Memory Allocation – Static vs Dynamic

- Using `malloc()` and `new()`
- Not always available
- Code can be large
- Is it thread safe?
- Is it deterministic?
- Can lead to fragmentation
- What to do if fails?
- Free RTOS (newer) allows static allocation at compile time



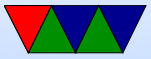
Is Regular Linux a RTOS

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



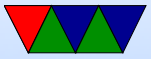
PREEMPT Kernel

- 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

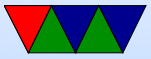


Typical kernel, when can you pre-empt

- 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

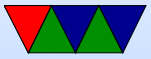


- Pre-empt patch makes it so almost any part of kernel can be stopped (pre-empted). Also moves interrupt routines into pre-emptible kernel threads.

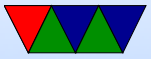


Linux PREEMPT Kernel

- 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)

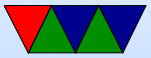


- Appears in ps as 0 to 139?



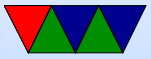
Changes to your code

- 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



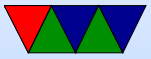
Interrupts

- 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



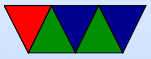
Co-operative real-time Linux

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

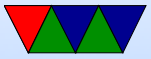


Non-Linux RTOSes

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