ECE 471 – Embedded Systems
Lecture 18

Vince Weaver

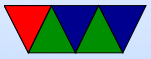
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

21 October 2024

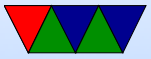
Announcements

- Don't forget RT homework HW#6
- HW#6 comment: the 3rd/4th experiment might slow down your system to be unusable if logged in at the GUI. If this happens and you can't complete the question, just mention that and complete things the best you can.
- RT related: SpaceX second stage trouble, engine burned 500ms more than should have



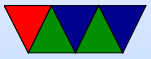
Notes on HW#4

- Please don't cheat
- Please try to get assignments in on time
- Don't ignore compiler warnings
- Took forever trying to grade switch debouncing, gave up in end, need some sort of automated test
- What you *definitely* shouldn't do is sleep for 100ms or more, that's long enough that a normal person pressing buttons will see lost keypresses



HW#5 – Code Notes – Datasheet

- What does 'X' mean in this context? (don't care)
- Bits 15-8 was confusing, it's because we can ignore bits 7-0 (the i2c address and r/w) as Linux sends those for us



HW#5 – Code Notes – Constants

- Enabling oscillator. If want value 2 in top 4 bits, 1 in bottom 4? Just use `0x21`?

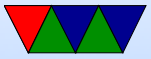
  `(0x2<<4)|(0x1)`

- Can we use hex or binary notation?

  The shifts make it more explicit what's going on, compiler will optimize for you

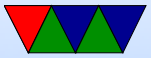
- "Magic Constants", you might instead want to do something like

```
#define HT16K33_OSCILLATOR_ON (0x2<<4)|(0x1)  // p42 of datasheet
buffer[0]=HT16K33_OSCILLATOR_ON;
```



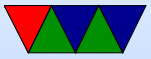
HW#5 Review – Questions – Why Use OS

- Why use OS?
- Why not?



HW#5 Review – Questions – i2c Reserved Address

- Skipped i2c – those addresses are reserved.
- For various things, not just “future purposes”
- What happens if you have a device living at address 0x0? Would it work?

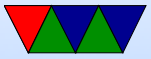


HW#5 Review – Linux

- wc, diff, piping

  `cat x.txt | sort | uniq | wc -l`
- You may have seen this all before in ECE331
- diff – used when making patches, also git diff

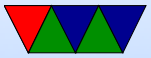
  Ask for wc -l which just shows lines. Can also show words, chars
- These days diff/patch more or less obsoleted by git pull requests



i2c Reserved Addresses Reminder

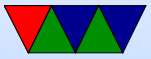
Address	R/W Bit	Description
000 0000	0	General call address
000 0000	1	START byte (helps make polling cheaper)
000 0001	X	CBUS address
000 0010	X	Reserved for different bus format
000 0011	X	Reserved for future purposes
000 01XX	X	Hs-mode master code
111 10XX	X	10-bit slave addressing
111 11XX	X	Reserved for future purposes

10-bit addresses work by using special address above with first 2 bits + R/W, then sending an additional byte with the lower 8 bits.



Operating System Scheduling

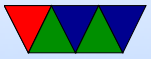
- It's been a while, but let's get back to talking about Real Time systems



Real Time without an O/S

Often an event loop. All parts have to be written so deadlines can be met. This means all tasks must carefully be written to not take too long, this can be extra work if one of the tasks is low-priority/not important

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



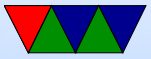
Real Time with an O/S

What if instead you ran all three at once, and let OS switch between them

```
while(1) {
  do_task1();
}
```

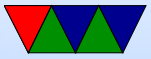
```
while(1) {
  do_task_2();
}
```

```
while(1) {
   do_task3();
}
```



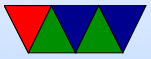
Bare Metal

- 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



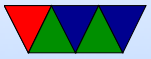
Common OS scheduling strategies

- Event driven – have priorities, highest priority pre-empts lower
  Usually can "yield" rest of your timeslice
- Time sharing – only switch at regular clock time, round-robin



Scheduler Types

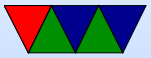
- There is a large body of work on scheduling algorithms.
- Assume you tell it to run tasks, they are put into queue
- How should they be run? A few (not exhaustive) possibilities:
  - Simple: In order the jobs arrive
  - Static: (RMS) Rate Monotonic Scheduling – shortest first
  - Dynamic: (EDF) Earliest deadline first



Deadline Scheduler Example

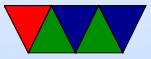
- Three tasks come in
  - A: deadline: finish by 10s, takes 4s to run
  - B: deadline: finish by 3s, takes 2s to run
  - C: deadline: finish by 5s, takes 1s to run
- Can they meet the deadline?

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	-	-	-



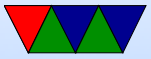
Where do the deadlines come from?

- Often from hardware, or from the spec
- Can you change them? Maybe, but might involve hardware changes
- Examples
  - Write to device, need to do all 4 memory accesses within 100us
  - Car notices brakes locking, start pumping them within 100ms
  - Drop laptop, notice and park hard-drive within 100ms

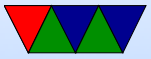


Priority Based Scheduling

- It's actually rare for an OS to let you specify a deadline
- Usually instead they are priority based (like in HW#6)
  - Have multiple tasks running, assign priority
  - In previous example, B highest, then C, then A
  - B can pre-empt C and A
- What can happen if overcommit resources? Starvation

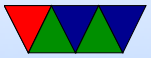


IRQ	-	-	-	-	-	I	-	-	-	-
HIGH	-	-	-	-	B	-	B	-	-	-
MEDIUM	-	-	C	-	-	-	-	-	-	-
LOW	A	A	-	A	-	-	-	A	-	-
OS	!	!	!	!	!	!	!	!	!	!



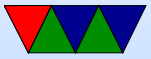
Priority Inversion

- 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.



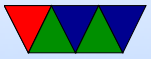
Real Time Operating System

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



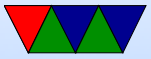
Hard Real Time Operating System

- 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



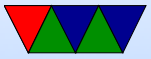
Priority Based, like Vxworks

- 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

- Useful article series about this: https://www.digikey.com/en/maker/projects/what-is-a-realtime-operating-system-rtos/28d8087f53844decafa5000d89608016
- Footprint as low as 9K
- Pre-emptive or co-op multitasking
- Regularly scheduled tasks (vTaskDelayUntil(), i.e. blink LED every 10 ticks)
- Task priority
- Semaphores/Mutexes
- Timers



- Stack overflow protection
- Inter-process communication (queues, etc)
- Power management support
- Interrupts (interrupt priority)