ECE 471 – Embedded Systems
Lecture 18

Vince Weaver

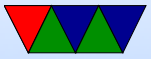
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

14 October 2020

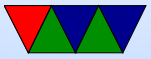
Announcements

- HW#6 still not posted yet. Soon.
- Midterm on Friday, the 16th



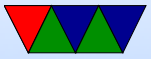
Midterm Notes

- Midterm will be *online* not in person
- If you do show up in person, bring a laptop as it will be electronic, not on paper.
- I will send a link on zoom and via e-mail right as class begins with a link to the exam
- I will be on zoom so I can answer questions. You do not need to log into zoom, but it might be useful in case you have questions.



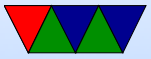
- It is open book/open notes, but *please* do not talk to anyone while taking the test

- The test will be short answer, similar to homeworks. Fill out a document and e-mail it at the end

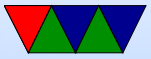


Midterm Content

- Be sure you know the four characteristics of an embedded system, and can make an argument about whether a system is one or not.
  - Inside of something (embedded)
  - Fixed-purpose
  - Resource constrained
  - Real time constraints (if you use this, be sure you understand)
- Benefits/downsides of using an operating system on an

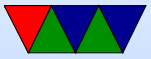


embedded device
  - Benefits: "Layer of Abstraction"
  - Downsides: overhead, timing
- C code
  - Have you look at some code and know what it is doing
  - Mostly know what file I/O, loops, usleep, open/ioctl (things we've done in the homeworks)
- Code Density
  - Why is dense code good in embedded systems?
- GPIO & i2c
  - Know some of its limitations (speeds, length of wires,



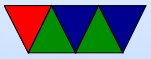
number of wires, etc)

- Don't need to know the raw protocol
- Know the Linux interface (open, ioctl, write) and be familiar with how those system calls work



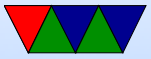
Some Last HW#4 Notes

- Comment your code!
- Note, you need to use the ioctl() to read values every time you want to read! Some code was reading once with the ioctl() and then accessing data.values[0] waiting for it to change. That value is just a plain integer, it's never going to change unless you read into it again.
- Debounce and sleep



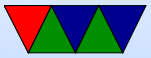
HW#5 Review – Questions

- Raspberry Pi boot odd: GPU does it. Why? Originally the chip was designed to be mostly GPU.
  sd-card is mildly unusual but not as unusual as GPU
- Fat32: gave lots of good reasons for Fat32, but the reason boot partitions often use it is it's simple enough to be read by firmware at extreme early boot. Q wasn't why FAT32 vs FAT16
- Program that loads kernel and jumps to it is called the bootloader



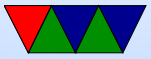
Not start.elf. Not an init script. Not the firmware.

- Skip i2c – those addresses are reserved.
  For various things, not just “future purposes”
  what happens if you have a device living at addr0?



HW#5 Review – Linux

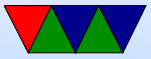
- wc, diff, piping
- You may have seen this all before in ECE331
- diff – used when making patches, also git diff
  Ask for wc -l which just shows lines. Can also show words, chars



i2c Reserved Addresses

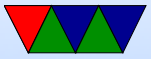
Address	R/W Bit	Description
000 0000	0	General call address
000 0000	1	START byte (helps make polling cheaper)
000 0001	X	CBUS address
000 0010	X	Reserved for different bus format
000 0011	X	Reserved for future purposes
000 01XX	X	Hs-mode master code
111 10XX	X	10-bit slave addressing
111 11XX	X	Reserved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



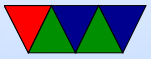
SMbus

- Enhanced i2c bus interface
- Has stricter rules about some signals
- Can do more advanced things, such as have slaves send notifications to master



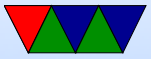
How a Program is Loaded on Linux

- Kernel Boots
- `init` started (systemd)
- `init` calls `fork()`
- child calls `exec()`
- Kernel checks if valid ELF. Passes to loader
- Loader loads it. Clears out BSS. Sets up stack. Jumps



to entry address (specified by executable)

- Program runs until complete.
- Parent process returned to if waiting. Otherwise, init.



Viewing Processes

- You can use `top` to see what processes are currently running
- Also `ps` but that's a bit harder to use.