ECE 471 – Embedded Systems Lecture 18

Vince Weaver

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

vincent.weaver@maine.edu

22 October 2018

Announcements

- HW#6 was posted, don't forget
- Info on project coming soon

Notes on HW#6

- Linking multiple C files together
- `gcc -c file.c` creates `file.o`
- Link a number of files together
  `gcc -o executable file1.o file2.o file3.o`
- All global functions/vars are exported, unless you declare them `static`
- How do you know how to call the functions?

Put prototypes in a header `.h` file

- This is how you create libraries (static just a matter of `ar`, dynamic are a bit more complicated)

HW#5 Question review

- Raspberry Pi boot odd: GPU does it. Why? Originally the chip was designed to be mostly 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?

- wc, diff, piping
  diff – used when making patches, also git diff
  Ask for wc -l which just shows lines. Can also show words, chars

Do a video game keyboard latency example

This is based on Dan Luu's Paper "Computer Latency: 1977-2017"

Real Time OS

Who uses realtime?

- Timing critical situations. Cars, medical equipment, space probes, etc.
- Industrial automation. SCADA. Stuxnet.
- Musicians, important to have low-latency when recording
- High-speed trading

Worst Case Behavior – Hardware

- Easier on older and simple hardware
- Old chips like 6502 – fixed clock, each instruction takes an exact number of cycles. Deterministic. With interrupts disabled you can perfectly predict how long code will take.

  Steve Wozniak famously wrote disk firmware on 6502 that more or less cycle-accurate bit-banged stepper motors.

  Also video games, racing the beam.

- Modern hardware more complex:
    - Memory accesses unpredictable with caches – may take 2 cycles or 1000 cycles
    - Interrupts can take unknown amount of time
    - Branch prediction
    - Power-save may change clock frequency
    - Even in manuals instructions can take a range of cycles

Software Worst Case – Context Switching

- OS provides the illusion of single-user system despite many processes running, by switching between them quickly.

- Switch rate in general 100Hz to 1000Hz, but can vary (and is configurable under Linux). Faster has high overhead but better responsiveness (guis, etc). Slower not good for interactive workloads but better for long-running batch jobs.

- You need to save register state. Can be slow, especially with lots of registers.
- When does context switch happen? Periodic timer interrupt. Certain syscalls (yield, sleep) when a process gives up its timeslice. When waiting on I/O
- Who decided who gets to run next? The scheduler.
- The scheduler is complex.
- Fair scheduling? If two users each have a process, who runs when? If one has 99 and one has 1, which runs

next?

- Linux scheduler was O(N). Then O(1). Now O(log N). Why not O(N^3)