ECE 471 – Embedded Systems
Lecture 15

Vince Weaver

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

`vincent.weaver@maine.edu`

10 October 2018

Announcements

- Midterm is Friday.

Homework #4 Review

- Still grading the code part.
- Questions
  - 5.a Why usleep? Less resources (not busy sleeping), cross-platform (not speed-of-machine-dependent), compiler won't remove, other things can run, power saving.
    Be careful saying accuracy! usleep() guarantees a minimum time delay, but it is best effort how long the delay actually is. So if you really need *exact* time

delays you probably want some other interface.

- 5.b Layer of abstraction. In this case, not having to bitbang the interface or know low-level addresses, portability among machines.
- 5.c Limitations : higher overhead, not all features exposed, uncertain timing.

  superuser permissions? when no OS you run everything as super user, though this depends on HW and is complicated.

  OS can't write as fast? OS has direct access to the hardware. It can write full bare-metal speed, the problem is you have to access through syscalls which can be slow.

- 5.d. Web browser part of OS? Microsoft law suit. Interesting comments on google/chrome
- 6.a Machines from dmesg: Pi3 (20) Pi3+ (8) pi2 (1) dmesg a good place to find error messages, etc.
- 6.b Kernel versions. Current Linus kernel (upstream) is 4.18/4.19-rc7
  Uname syscall, what the parts mean

```
Linux linpack-test 4.14.50-v7+ #1122 SMP Tue Jun 19 12:26:26 BST 2018 armv7l GNU/Linux\\
Linux orvavista 4.5.0-2-amd64 #1 SMP Debian 4.5.5-1 (2016-05-29) x86_64 GNU/Linux\\
```

2018: 4.4.50 (1) 4.4.70 (1) 4.9.14 (1) 4.9.35 (3)

4.9.41 (1) 4.9.59(2) 4.9.80 (1) 4.14.34 (1) 4.14.50 (6) 4.14.52(2) 4.14.60 (1) 4.14.62 (1) 4.14.69 (4) 4.14.70 (2)

- 6.c. Disk space. Why `-h`? Human readable. what does that mean? Why is it not the default? At least Linux defaults to 1kB blocks (UNIX was 512) Lots of large disks.

Midterm Review

- 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
  - Cost, time to market, helper libraries, overhead, timing
- C code
  - Have you look at some code and know what it is doing
  - Mostly know what file I/O, loops, and string maninpulations work (things we've done in the homeworks)
- Code Density
  - Why is dense code good in embedded systems?
  - What changes were needed to ARM32 to make it fit into 16-bit THUMB?

- 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

Booting a System

Firmware

- What is firmware?

Device Firmware

- Devices are their own embedded systems these days. May even have full CPUs, etc.
- Need to run code. Firmware.
- In ROM? Or upgradable? Why might you want to upgrade? (bug fixes, economy, etc.)
- Talk about recent USB firmware malware

Firmware

Provides booting, configuration/setup, sometimes provides rudimentary hardware access routines.
Kernel developers like to complain about firmware authors.
Often mysterious bugs, only tested under Windows, etc.

- BIOS – legacy 16-bit interface on x86 machines
- UEFI – Unified Extensible Firmware Interface
  ia64, x86, ARM. From Intel. Replaces BIOS
- OpenFirmware – old macs, SPARC
- LinuxBIOS

Boot Methods

Firmware can be quite complex.

- Floppy
- Hard-drive (PATA/SATA/SCSI/RAID)
- CD/DVD
- USB
- Network (PXE/tftp)

- Flash, SD card
- Tape
- Networked tape
- Paper tape? Front-panel switches?

Bootloaders on ARM

- uBoot – Universal Bootloader, for ARM and under embedded systems
- So both BIOS and bootloader like minimal OSes

Raspberry Pi Booting

- Unusual
- Small amount of firmware on SoC
- ARM 1176 brought up inactive (in reset)
- Videocore loads first stage from ROM
- This reads `bootcode.bin` from fat partition on SD card into L2 cache. It's actually a RTOS (real time OS in own right "ThreadX")

- This runs on videocard, enables SDRAM, then loads `start.elf`

- This initializes things, the loads and boots Linux `kernel.img`. (also reads some config files there first)

Bootloaders on ARM

- uBoot – Universal Bootloader, for ARM and under embedded systems
- So both BIOS and bootloader like minimal OSes