ECE 471 – Embedded Systems
Lecture 15

Vince Weaver

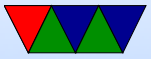
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

7 October 2019

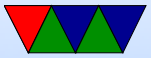
Announcements

- HW#5 was posted
- Midterm is a week from Friday, the 18th

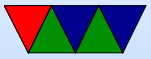


Homework #4 Error Checking

- What do you do if there's an error?
- Ignore it? Why could that be bad?
- Retry until it succeeds?
- Print an error message and continue?
  Can you continue?
  What if continuing with a bad file descriptor breaks things?
  What if printing too many error messages fills up a log, swamps the screen, hides other errors?

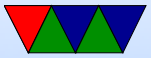


- Good error message
  Can't be confused with valid input (airlock)
  If displayed to user, make it easy to understand
- Print an error message and exit?
  What if it's a critical system?
- Crashing is almost never the right answer.

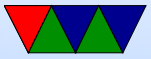


Homework #4 Permissions

- We haven't really discussed Linux permissions
- List file, "user" "group" "all"
- `drwxr-xr-x`
- Often in octal, 777 means everyone access
- Devices under /dev or /sysfs might be set to only root or superuser
- Traditional UNIX /dev you can set with chown (to set user/group) or chmod (to set permissions)
- Group under /etc/group, so gpio group



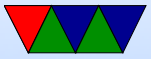
- Having to manually set permissions a pain. Program called udev that does it automatically when a device driver is configured. It might take a few ms to notice
- Why is it better than using "sudo"? Why might I as grader not want to run your code using "sudo" if I can avoid it?
- How to set up sudo? /etc/sudoers file



Homework #4 Review

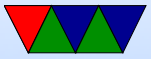
- Blink frequency. Remember, 1Hz is 500ms on / 500ms off

  not 500us, not 1s
- Still grading the switch part.



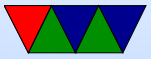
Homework #4 Question

- 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.
ability to run WiringPi is not a benefit

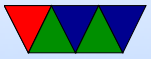
- 5.c Limitations : higher overhead, not all features exposed, uncertain timing.
  superuser permissions? when no OS you run everything as super user, though this depends on HW and is complicated.
  OS can't write as fast? OS has direct access to the hardware. It can write full bare-metal speed, the problem is you have to access through syscalls which can be slow.



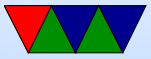
- 6.a Machines from dmesg: Pi4 (3) Pi3 (8) pi3b+ (8) pi2 (1) dmesg a good place to find error messages, etc.
- 6.b Kernel versions. Current Linus kernel (upstream) is 5.3 /5.4-rc2
  Uname syscall, what the parts mean

```
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\\
```

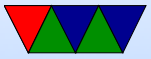
  2019: 4.4.11 (1) 4.14.79 (3) 4.14.98 (1) 4.18.57 (1) 4.19.42 (1) 4.19.57 (4) 4.19.66(9)
- 6.c. Disk space. Why -h? Human readable. what does that mean? Why is it not the default? At least Linux defaults to 1kB blocks (UNIX was 512) Lots of large



disks.

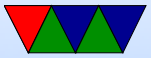


Booting a System



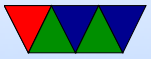
Firmware

- What is firmware?



Device Firmware

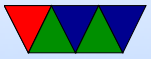
- Devices are their own embedded systems these days. May even have full CPUs, etc.
- Need to run code. Firmware.
- In ROM? Or upgradable? Why might you want to upgrade? (bug fixes, economy, etc.)
- Talk about recent USB firmware malware



Firmware

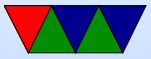
Provides booting, configuration/setup, sometimes provides rudimentary hardware access routines.
Kernel developers like to complain about firmware authors.
Often mysterious bugs, only tested under Windows, etc.

- 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



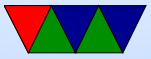
Bootloaders

- Firmware doesn't usually directly load Operating System
- Bootloader (relatively simple code, just smart enough to load OS and jumpt to it) is loaded first
- Bootloader is often on a very simple filesystem (such as FAT) as the code has to be simple (possibly even written in assembly language)
- Bootloader is often just complex enough to load OS kernel from disk/network/etc and jump to it



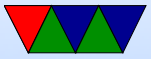
Raspberry Pi Booting

- 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