ECE 471 – Embedded Systems
Lecture 11

Vince Weaver

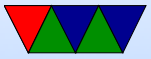
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

7 October 2014

Announcements

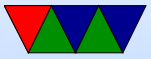
- Midterm is tentatively scheduled for October 21st.



Homework #4 Review

- Submit the right files!
  Make sure the code compiles!
  Include your name!

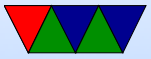
- Review of how `read()` and `write()` work, the main issue being you read into a buffer pointed to by a pointer, and it's important to get the number of bytes read/write right.
  For `open()` need proper O_RDONLY, O_WRONLY
  General C stuff. Try to fix compiler warnings.



Control-C and what it does.
void functions, pre-declare functions
Error checking! Silent fail is bad.

- For the switch question, wanted it to print on press and on release. Have a state bit that remembers last state. Remember the value you read is the ASCII value of result.

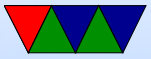
- Debounce
  Was it needed in practice? Comment your code if



debounce included! One way is to when changes, sleep a small amount (1ms?) and re-read and see if the value is the same. Keep reading until get same twice in a row. Also saw just once it changes, sleep some, in the assumption the first change is right and other spurious.

- 5.a Why udelay? Less resources (not busy sleeping), cross-platform, compiler won't remove, other things can run

- 5.b Layer of abstraction. In this case, not having

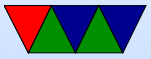


to bitbang the interface or know low-level addresses, portability among machines.

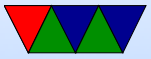
- 5.c Limitations : higher overhead, not all features exposed, uncertain timing

- 5.d. Web browser part of OS? Microsoft law suit. Interesting comments on google/chrome

- 6. Machine BCM2708 (not BCM2835)
  dmesg good place to find if hardware working, error messages, etc.
  b. 3.10.25+ (1)

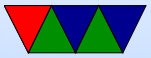


3.2.27 (1)
3.6.11+ (1)
3.8.13 (1 beaglebone)
3.10.30 (3)
3.12.22+ (3)
3.12.28+ (5) (me)
3.12.33+ (1)
c. Disk space. Why `-h`? Human readable.

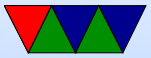


Booting Linux

- Bootloader jumps into OS entry point
- Set Up Virtual Memory
- Setup Interrupts
- Detect Hardware / Install Device Drivers
- Mount filesystems
- Pass control to userspace / call init

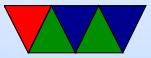


- Run init scripts
- rc boot scripts, /etc/rc.local
  Start servers, or “daemons” as they’re called under Linux.
- fork()/exec(), run login, run shell



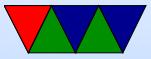
How a Program is Loaded on Linux

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



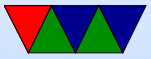
to entry address (specified by executable)

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



Viewing Processes

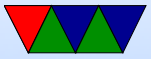
- You can use `top` to see what processes are currently running
- Also `ps` but that's a bit harder to use.



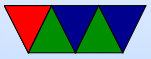
Context Switching

- 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?

- Various O() ratios for the schedulers