ECE 471 – Embedded Systems
Lecture 8

Vince Weaver

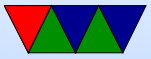
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

26 September 2013

Announcements

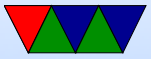
- HW#2 is delayed
- Read chapter 11 in textbook



Brief Overview of the Gumstix Overo Board

TODO: Put a diagram here

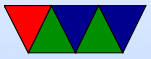
More details after boards are distributed.



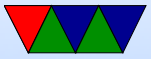
Coding Directly for the Hardware

One way of developing embedded systems is coding to the raw hardware, as you did with the STM Discovery Boards in ECE271.

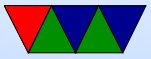
- Compile code
- Prepare for upload (hexbin?)
- Upload into FLASH
- Boots to offset



- Setup, flat memory (usually), stack at top, code near bottom, IRQ vectors
- Handle Interrupts
- Must do I/O directly (no drivers)
  Although if lucky, can find existing code.

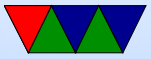


Instead, one can use an Operating System

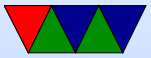


Why Use an Operating System?

- Provides Layers of Abstraction
  - Abstract hardware: hide hardware differences. same hardware interface for classes of hardware (things like video cameras, disks, keyboards, etc) despite differing implementation details
  - Abstract software: with VM get linear address space, same system calls on all systems
  - Abstraction comes at a cost. Higher overhead, unknown timing

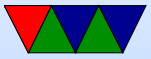


- Multi-tasking / Multi-user
- Security, permissions (Linus dial out onto /dev/hda)
- Common code in kernel and libraries, no need to re-invent



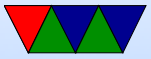
What's included with an OS

- kernel / drivers – Linux definition
- also system libraries – Solaris definition
- low-level utils / software / GUI – Windows definition
  Web Browser included?
- Linux usually makes distinction between the OS Kernel and distribution. OSX/Windows usually doesn't.



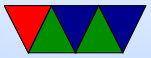
Operating Systems Types

- Monolithic kernel – everything in one big address space. Something goes wrong, lose it all. Faster
- Microkernel – separate parts that communicate by message passing. can restart independently. Slower.
- Microkernels were supposed to take over the world. Didn't happen. (GNU Hurd?)
- Famous Torvalds (Linux) vs Tannenbaum (Minix) flamewar

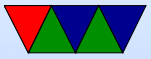


Common Desktop/Server Operating Systems

- Windows
- OSX
- Linux
- FreeBSD / NetBSD / OpenBSD
- UNIX (Irix/Solaris/AIX/etc.)

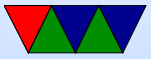


- BeOS/Haiku

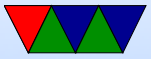


Embedded Operating Systems

- Microsoft WinCE, Windows Mobile
- Linux / Android
- VXworks – realtime OS, used on many space probes
- Apple iOS
- QNX – realtime microkernel UNIX-like OS, owned by Blackberry now

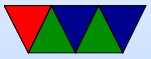


- Cisco iOS

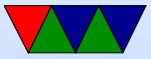


Embedded Linux Distributions

- linaro – consortium that work on ARM software
- openwrt – small distro initially designed for wireless routers
- yocto – Linux Foundation sponsored embedded distro
- maemo – embedded distro originally by Nokia (obsolete)
- MeeGo – continuation of maemo, also obsolete

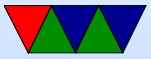


- Tizen – Follow up on MeeGo, by Samsung and Intel
- Ängstrom – Merger of various projects
- And many others. It's very easy to put together a Linux distribution

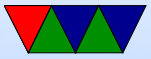


Linux/UNIX History

- UNIX invented early 70s at Bell Labs
- Widely distributed by academics
- Berkeley makes their own BSD version
- By the 90s many companies selling UNIX workstations. Expensive.
- Linus Torvalds in 1991 wanted own UNIX-like OS. Minix (which he used for development) limited to academic use

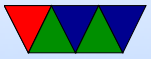


and non-free. The various BSDs caught up in lawsuit with AT&T. So he wrote his own.



Licensing and its Effects

- Linux under GPLv2.
- The Free Software Foundation has moved most of its software (including gcc compiler) to the less popular GPLv3 which most companies don't like.
- Companies often prefer BSD type license which has fewer restrictions; companies can use code and release binaries without having to release the source (a GPL requirement).



- Apple and Google both trying to replace as much code as possible with BSD versions.

- Linux popular in embedded space because it is cheap/free and source code is available.