ECE 598 – Advanced Operating Systems
Lecture 1

Vince Weaver

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

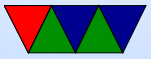
vincent.weaver@maine.edu

23 January 2018

Announcements

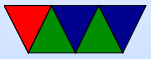
- Distribute and go over syllabus

  `http://web.eece.maine.edu/~vweaver/classes/ece598_2018s/ece598_2018s.pdf`

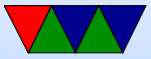


Syllabus Summary

- Does require some C and low-level Assembly. For the non-computer engineers will try to go over it as much as possible.
- Will involve setting up an ARM toolchain (possibly cross-compiler) that also can be tricky at first.
- There will be some manner of low-level serial port access which is hard at first.
- Please, please, no academic dishonesty.

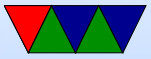


- There will be a final project
- If you need help on homeworks, come to me sooner rather than later.



Hardware for the Class Assignments

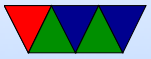
- Raspberry Pi Model 2 or 3.
  We need multicore, so no B+, B, A, A+, or zero.
  I can potentially loan one out if you do not have one.
- Micro-USB cable (To provide power)
- 4GB SD memory card (or larger) – preloaded with Linux if possible
- USB/Serial adapter – something similar to
  `http://www.adafruit.com/products/954`
  I will loan these out, no need to buy one



- Some way to write an SD-card

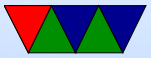
If you are starting from scratch, getting a kit like `http://www.adafruit.com/products/1014` might be convenient (but expensive) getting the parts individually might be cheaper.

I have a limited number of Raspberry Pi boards I can loan out if necessary.



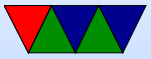
Optional Hardware (not necessary for class but might make development easier)

- Case to hold the Pi
- GPIO breakout cable
- ethernet cable
- HDMI cable
- USB keyboard

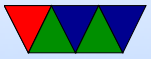


Why Use an Operating System?

- 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

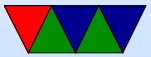


- Multi-tasking / Multi-user – why useful?
- Security, permissions (Linus dial out onto /dev/hda)
- Common code in kernel and libraries, no need to re-invent

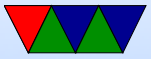


Common Operating Systems

- UNIX-like – UNIX (Solaris, IRIX, AIX, ULTRIX, XENIX), Linux, FreeBSD, OpenBSD, NetBSD, OSX/ioS, MINIX
- VMS
- WindowsNT based (NT/2000/XP/Vista/8/10/etc)
- CP/M, DOS based (DOS, Windows 3.1, Windows 95/98/ME)
- Embedded OSes (QNX, Vxworks, OpenRTS, ThreadX)

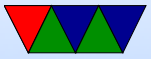


- Mainframe OSes (IBM z/OS)
- Other – MacOS, BeOS, AmigaOS, Haiku, Plan9



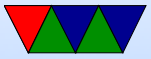
In this class will primarily discuss Linux

- Free
- Source code available
- I know it well; have contributed many patches



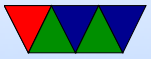
What's included with an OS

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



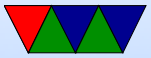
Linux Distributions

- RedHat/Fedora/Suse/Ubuntu/Debian



What Does Linux Provide

- Boot/initialization
- Hardware drivers
- Network (TCP/IP and others)
- Interrupts, DMA
- Multi-tasking/Job scheduling
- Virtual Memory
- Filesystems
- Security



What Language do you write OS in?

- Assembly Language?
- C?
- C++?
- Java? Python? Javascript?
- Rust? Go?