ECE 531 – Advanced Operating Systems
Lecture 1

Vince Weaver

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

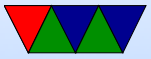
vincent.weaver@maine.edu

3 September 2025

Welcome to ECE531

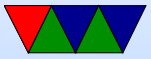
The class COS doesn't want you to take!

We're going to learn more about Operating Systems



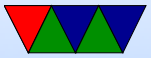
Reviewing the Syllabus

https://web.eece.maine.edu/~vweaver/classes/ece531_2025f/ece531_2025f.pdf



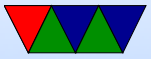
Syllabus – Instructor Info

- Office is 203 Barrows
- Tentative Office hours 11:30am-12:30pm Tues/Thurs. Feel free to stop by if door open



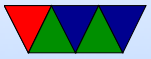
Pre-reqs / Requirements

- ECE331/ECE471 or equivalent experience
  - Some previous Linux knowledge helps
  - 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.



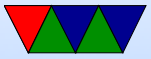
Textbook

- No required textbook.
- A few recommended books if you would like a reference.



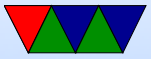
Syllabus – Hardware

- We will use Raspberry Pis. More on that later.

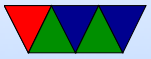


Syllabus – grading

- Homeworks, 50%: 10-11 total, lowest dropped.
  - Generally will be due on Friday by beginning of class. Will have a week to do them.
  - Submission by e-mail, grades sent in response to that e-mail, if you don't like that let me know.
  - Will send e-mail when assignment posted on website.
  - Will reply with grades. No Brightspace?
- Midterms, two, 25% total
  Tentatively 20 October and 5 December

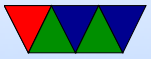


- No final (note Mainestreet has one listed)
- Class participation, 5%
  Part of this is returning borrowed items at end.
- Project, 20%: Involves using what you learned to do an operating-system project, with a final writeup and demo the last week of classes. Can work in group. More details as we get closer.



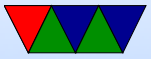
Syllabus – Late Work / Regrade

- Late work penalty. I will consider late work, but best to turn in what you have at time.
- Make regrade requests via e-mail.



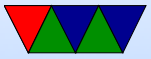
Homework Help

- I'll be glad to help if you get really stuck on homeworks
- Often the easiest way to do this is send me your code, as I can run it through the compiler and test it. Describing your issue or sending me a screenshot might not be enough and I'll probably ask you to send your code



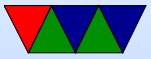
Covid/Mask Policy

- Follow UMaine Guidance
- If you test positive for Covid please don't come to class
- If you are sick for any reason but still coming to class I encourage you to wear a mask



Syllabus – Academic Honesty

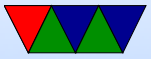
- This has been a problem in the past!
- Do not copy code from other students, either current or from previous years.
- Asking help from the professor/TA is fine
- Asking for general help, or discussing with classmates is fine
- Even having someone look over your code to help find a problem is fine
- Try to avoid giving someone code to use as a reference



as in my experience it's too tempting and the person will "accidentally" submit it as their own

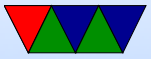
- Just don't copy someone else's code and submit it as your own

  This includes cut-and-paste or retyping
- Also don't copy code off the internet (again, looking for advice online is fine, but copying code directly is not)
- Don't use AI tools that do the homework for you! (Like Microsoft/Github Co-pilot/ChatGPT)
- If caught copying, you will get a 0 on the assignment and so will the person who provided the code.



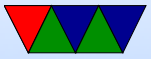
Hardware for the Class Assignments

- I will loan out hardware
  - Raspberry Pi Model 1B+
  - 4GB or larger micro-SD card
  - USB/Serial adapter – something similar to `http://www.adafruit.com/products/954`
- Ideally you'll have your own way to write an SD-card but I have some adapters too



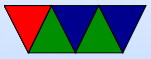
Why Raspberry Pis? And Specifically 1B+

- Pi is real hardware (not a simulator!), relatively cheap, and relatively well documented
- The problem is each model of Pi has very different hardware
- Different chips, newer ones are 64-bit multicore, various low-level things change each release. It's hard to keep up.
- Even just blinking the ACT LED changes each release

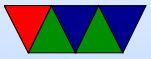


Class Plan

- We'll learn about modern operating systems principles, primarily using Linux for examples
- We'll also have assignments where we write our own, custom, OS that runs on a Raspberry Pi



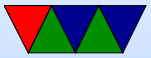
The Case for Operating Systems



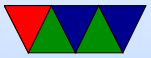
Coding Directly for the Hardware

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

- Compile code
- Prepare for upload (hexbin?)
- Upload into FLASH
- Boots to offset (jumps to interrupt vector on RESET)
- Setup, flat memory (usually), stack at top, code near bottom, IRQ vectors

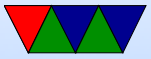


- Handle Interrupts
- Must do I/O directly (no drivers)
  Although if lucky, can find existing code.
- **Code is specific to the hardware you are on**



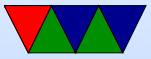
Problems with "Bare Metal"

- It is difficult and low-level
- Why not offload the tricky stuff to code written by someone else?
- These days usually that's an Operating System
- Although in this class, *we're* the someone else

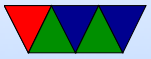


Why Use an Operating System?

- 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
- Other benefits:
  - Multi-tasking / Multi-user

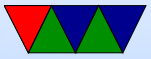


- Security, permissions (Linus dial out onto /dev/hda)
- Common code in kernel and libraries, no need to re-invent
- Handle complex low-level tasks (interrupts, DMA, task-switching)



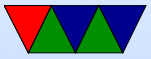
Downsides of Operating System?

- Overhead / Abstraction has a cost
  - Higher overhead (speed)
  - Higher overhead (memory)
  - Unknown timing (Real Time)
- Security
  - Larger code base can provide larger attack surface



Common Desktop/Server Operating Systems

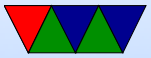
- UNIX derived
  - Linux (clone imlpemented from scratch)
  - FreeBSD / NetBSD / OpenBSD
  - MacOS (FreeBSD/Nextstep heritage)
  - Legacy (Irix/Solaris/AIX/ULTRIX/XENIXetc.)
- WindowsNT (NT/2000/XP/Vista/8/10/11)
- CP/M, DOS based (DOS, Windows 3.1, 95/98/ME)
- Obscure: BeOS/Haiku, hurd, RiscOS



Embedded Operating Systems

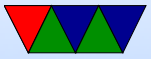
- Cellphone/Tablet
  - Android (Linux)
  - ChromeOS (Linux)
  - Apple iOS
  - Microsoft (WinCE/Mobile/Phone/RT/S/IoT (all these have been discontinued))

    In theory can install Windows 11 on a Raspberry Pi
- Networking
  - OpenWRT (Linux)



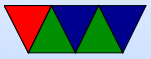
  - Cisco iOS
- Real Time OS
  - VXworks – realtime OS, used on many space probes
  - QNX – realtime microkernel UNIX-like OS, owned by Blackberry now
  - ThreadX – found in Pi GPU, Microsoft owns now?

    `https://www.theregister.com/2023/11/28/microsoft_opens_sources_threadx/`
  - FreeRTOS



We will work on our own OS

- Use a variant of “vmwOS”, an Operating System I wrote
- Other educational OSes exist, such as MINIX and xv6
- The fun part here is it will run on actual Raspberry Pi hardware, not in a simulator or emulator



We will also learn about mainstream OSes

- Concentrate on Linux
- Free
- Source code available
- I know it well; have contributed many patches
- It is showing its age though, not as exciting to work on as in 90s
- What will replace Linux?