ECE 531 – Advanced Operating Systems
Lecture 1

Vince Weaver

https://web.eece.maine.edu/~vweaver
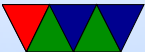
vincent.weaver@maine.edu

29 August 2023

Welcome to ECE531/598!

We're going to learn more about Operating Systems

Reviewing the Syllabus

https://web.eece.maine.edu/~vweaver/classes/ece531_2023f/ece531_2023f.pdf

Syllabus – Instructor Info

- Office is 203 Barrows
- Tentative Office hours 11am-noon Wed/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

- You will need a Raspberry Pi. More on that later.

Syllabus – grading

- Homeworks, 50%: 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. Brightspace?
- Midterms, two, 25% total
  Tentatively 17 October and 28 November

- No final
- 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
- I'm still not convinced we're in the clear, but everyone else has given up
- If you test positive for Covid please don't come to class and let me know and we can make sure you get the work done
- 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)

Hardware for the Class Assignments

- Raspberry Pi Model 2 or 3.
  Still working on 4 support
  Will try to let you know soon as possible
- 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

Why Pi's?

- 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

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

The Case for Operating Systems

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.
- **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
  - 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 (note: while this can be a feature of an OS, it's not exclusive to that. You can have code libraries for bare-metal systems)
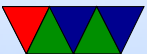
Common Operating Systems

- 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

- It is showing its age though, not as exciting to work on
  as in 90s

- What will replace Linux?

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(?)
- Graphics?

What Language do you write OS in?

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