cs5965 Advanced OS Implementation

Lecture 00 – Logistics

Anton Burtsev

Class details

- Mostly graduate class but undergrads are welcome to take
  - 20 people
- Prerequisites
  - Xv6-based cs5460
- Instructor
  - Anton Burtsev
  - https://mars-research.github.io/aburtsev/

- Class web page
  - https://mars-research.github.io/cs5965/

Some background info on me

- I build operating systems since 1999?
  - E1 OS
    - Distributed OS as a student project (naïve)
  - micro-ITRON OS on top of L4
    - Internship with Gernot Heiser and L4 team at UNSW
  - XenTT
    - Deterministic replay for Xen VMM
  - LXDs/LVDs/Ksplit
    - Device driver isolation in the Linux kernel
  - RedLeaf
    - Clean slate OS in Rust
  - Atmosphere
    - Formally verified microkernel in Rust and Verus

This course: Build your own OS

- Each of you will build a small but functional microkernel
  - Ideally in Rust
  - C is possible as an alternative
- Specifically
  - Boot into Rust
  - Create an address space
  - Load an ELF into that address space (hello from user)
  - Timer and other interrupts
  - Context switch
  - Inter-process communication (IPC)
  - If time allows a user-level device driver

This course: Build your own OS

- Lectures
  - I explain a high-level idea of the next step
    - E.g., how to boot into Rust
  - We possibly go through some source code that does it
  - You build your implementation as a homework
- I also explain how different tools work
  - E.g., QEMU emulator and how to use it for OS development
- We read and discuss relevant papers
  - To get a better hold of possible design choices

Grading

- No midterm, no final
- Homework submissions (90%)
- In-class participation (10%)

Plagiarism

- All work should be your own
- Feel free to discuss approaches, show your code, etc.
- But in the end you have to type your own code and pass the tests

AI use policy

- AI is allowed as a tool to automate mundane work
  - E.g., write a page table initialization loop in Rust

Thank you

Questions