EECS 678: Introduction to Operating Systems

Heechul Yun
About Me

• Heechul Yun, Assistant Prof., Dept. of EECS
  – Office: 3040 Eaton, 236 Nichols
  – Email: heechul.yun@ku.edu

• Research Areas
  – Operating systems and architecture support for embedded/real-time systems
    • To improve time predictability, energy efficiency, and throughput
    • Multicore, memory systems

• Previously
  – Worked as a systems software engineer at Samsung Electronics
    • mainly worked on Linux kernel

• More Information
  – http://ittc.ku.edu/~heechul
About This Class

• Objectives: Learn OS basics and practical system programming skills
• Prerequisite: C language
• Textbook: Operating System Concepts
• Audience: Senior and junior undergraduate (grad students)
• Course website: http://ittc.ku.edu/~heechul/courses/eecs678/
Course Structure

• Lectures
  – TR 8:00 – 9:15 @LEA 2112
  – Office hour: TR: 9:20 - 10:00 a.m. @ 3040 Eaton
  – Discuss OS concepts and the design of major OS components

• Quiz/Homework
  – Occasional online quizzes to check your understanding

• Lab
  – Tue 4:00 - 5:50 p.m., Fri 9:00 – 10:50 p.m. @1005B Eaton
  – Hands-on system programming experiences.
  – Each lab includes lab discussion and an assignment

• Programming projects
  – Design and implement some parts of OS.
  – 3 projects: 1) Shell, 2) CPU scheduler, 3) Memory allocator (2week/prj)
  – To do in groups of two persons. Solo project is also allowed.
Grading

- Class Participation: 5%
- Exam: 45% (Mid:20%, Final:25%)
- Quiz/Homework: 5%
- Lab: 15%
- Projects: 30%
- Bonus points: up to 5%
  - Active class participation
  - Extra features in programming projects
Policy

• Late submissions
  – Lab assignments: **not allowed.**
  – Projects: 20% off each additional 24 hours delay (~24h = 80%, ~48h = 60%, ~72h=40%, ~96h=20%, >96h = 0%)

• Cheating
  – You can discuss about code and help find bugs of your peers. However, copying another’s code or writing code for someone else is cheating and, if identified, the involved students will be notified to the department chair
Operating Systems Are Everywhere

• Computers
• Smart phones
• Cars
• Airplanes
• ...
• Almost everything
What is an Operating System?
What is an Operating System?

- A program that acts as an intermediary between users and the computer hardware.
What is an Operating System?

• An easy to use virtual machine – **User’s view**
  – Hide complex details for you.
    • What CPU am I using? Intel or AMD?
    • How much memory do I have?
    • Where and how to store my data on the disk?

  – Provide APIs and services
    • read(...), write(..)
    • Virtual memory, filesystems, ...
What is an Operating System?

• A resource manager – *System’s view*
  
  – Make everybody get a fair share of resources
  
  • Time and space multiplexing hardware resources
  
  – Monitor/prevent error or improper use
What is an Operating System?

• Is an internet browser part of an OS?
  – Everything that shipped by the OS vendor?
  – What about ‘solitaire’?

• The program that always runs
  – Typically in kernel mode (we will learn it later)
Why Needed?

• Programmability
  – You don’t need to know hardware details to do stuffs

• Portability
  – You can run the same program on different hardware configurations

• Safety
  – The OS protects your program from faults in other programs

• Efficiency
  – Multiple programs/users can share the same hardware efficiently
What to Study?

• Not “how to use”
  – I’m sure you know better than me about how to use, for example, the iOS in your iPhone.

• But “how it works!”
  – We will study underlying concepts, standard OS components and their designs
OS Design Issues

• Structure
  – How to organize the OS?
• Communication
  – How to exchange data among different programs?
• Performance
  – How to maximize/guarantee performance and fairness?
• Naming
  – How to name/access resources?
• Protection
  – How to protect with each other?
• Security
  – How to prevent unauthorized access?
• Reliability
  – How to prevent system crash?
Why Study?

• I’m a user
  – Have you ever wondered how it works?
  – You can better tune the OS to improve performance (or save energy)

• I’m a system programmer
  – You can write more efficient programs by knowing how the OS works.

• I’m a hacker
  – You need to know the enemy (the OS) to beat it
Brief History of Computers

• Early computing machines
  – Babbage’s analytical engine
  – First programmer: Ada Lovelace

• Vacuum tube machines
  – 1940s ~ 1950s
  – Used to break code in WWII
  – No OS, No PL
Brief History of Computers

• Vacuum tubes $\rightarrow$ Transistors $\rightarrow$ IC $\rightarrow$ VLSI
  - Smaller, faster, and more reliable
  - Enable smaller computers
• 1960s Mainframes
• 1970s Minicomputers
• 1980s Microprocessor, Apple, IBM PC
• 1990s PC, Internet
• 2000s Cloud computing
• 2010s Mobile, Internet-of-things (IoT)
Evolution of Operating Systems

• Batch systems
  – Each user submits her job on punch cards
  – Collect a batch of jobs, read the batch before start processing
  – The ‘OS’ processes each job at a time
  – Problems
    • No interactivity
    • CPU is underutilize to wait I/O operations

IBM 029 card punch

http://www.catb.org/esr/writings/taouu/html/ch02s01.html
Evolution of Operating Systems

• Multiprogramming
  – Multiple runnable jobs at a time
  – I/O and compute can overlap
  – OS goal: maximize system throughput
  – IBM OS/360
Evolution of Operating Systems

• Timesharing
  – Multiple interactive users sharing a machine
  – Each user accesses the machine via a terminal
  – Provide each user an illusion of using the entire machine
  – OS goal: optimize response time
  – UNIX
Evolution of Operating Systems

• Parallel computing
  – Use multiple CPUs/cores to speed up performance
  – OS goal: fast synchronization, max utilization

• Distributed computing
  – Physically separate networked computers

• Virtualization
  – Multiple OSes on a single machine
Challenges for Future OS

• New kinds of hardware are keep coming
  – Heterogeneous multicore processors (e.g., ARM big.LITTLE)
  – Storage Class Memory (SCM): non-volatile DRAM-like memories

• New computing paradigms
  – Cloud computing
  – Internet-of-Things (IoT)
Summary

• In this class, you will learn
  – Major OS components
  – Their structure, interface, mechanisms, policies, and algorithms

• This class will (hopefully) help you
  – Understand the foundation of computing systems
  – Understand various engineering trade-offs in designing complex systems you would build in future
Prospective Students

- Intelligent vehicles
  - ROS, python, Linux, OpenCV, CUDA
  - Intelligence and perception
    - E.g., Vision based navigation
- OS and architecture
  - Open-source hardware and Linux
  - Hardware/OS extensions for CPS
- Send me your CV and schedule a meeting

https://riscv.org/