ICS 612
Theory of Operating Systems

Edo Biagioni
esb@hawaii.edu

Information and Computer Sciences
University of Hawaii at Mānoa

Today's plan

- course overview
- why study operating systems?
- theory and implementation
- a range of operating systems
- historical background

Course Details

- esb@hawaii.edu
- http://www2.hawaii.edu/~esb
- Lectures are Tuesdays and Thursdays, 12noon in Holmes 242
- textbook: "*Operating Systems Design and Implementation*", **Third** Edition, Andrew Tanenbaum, Albert Woodhull, (Prentice-Hall).
- *n* projects (there were 7 projects in 2011), 2 exams and a final, one presentation
- office hours (*beginning next week*): Tuesdays 10-11 and 3-4, Thursdays 3-4. Also by appointment, by email or phone, or when my door is open and I am not otherwise busy.

Course Contents

- much flexibility, what follows is tentative
- hands-on operating system development: Minix
- operating system fundamentals: processes, storage, resource management, security
- varieties of operating systems:
  - distributed: network file systems, multiprocessors, grid computing
  - embedded (TinyOS)
  - historical and current: Multics, Unix, Linux, seL4, mobile OSs
  - virtual machines and containers
- device drivers
- shells
- compared to ICS 332, much of the same material, but much greater depth, more hands-on experimentation, different systems, a greater range of systems studied, and student presentations

Why Study Operating Systems?

- might have to design or build parts of them yourself: threading systems, distributed applications, synchronization, security
- might want to use them to full advantage, e.g. tuning and customizing, selecting an appropriate product
- might find operating systems interesting
- general foundation, general knowledge, or maybe Ph.D. qualifying exam

What is (not) covered in this course

- A **system administrator** manages a machine and configures an operating system, matching requirements to capabilities.
  - This course may or may not help you become a better system administrator:

    if you already are a good system administrator, it may make you a better system administrator, but it will not cover the nuts and bolts of system adminstration
- An **operating system designer** develops a set of requirements and transforms a set of requirements into a design
- An **operating system developer** transforms a design into an operating system, or maintains an operating system by adding **device drivers** or **fixes**
- An **application developer** should know what to expect of an operating system, and may have to implement some OS-like features
- this course is designed to present information needed by operating system designers and application and operating system developers

Theory

- deadlock can be very complicated:
  - deadlock prevention
  - deadlock detection
  - deadlock avoidance
  - deadlock resolution
- many interesting solutions have been proposed
- but in most cases, only deadlock prevention is useful and practical
- appropriate theory enhances understanding and aids in designing
- impractical theory ("theoretical" theory) is interesting, but not as useful, can illustrate why the practical choices are more effective
- this course: theory and practice

Implementation

- Minix: a real operating system designed for the study of operating systems
  - eventually re-designed and re-implemented and much expanded by Linus Thorvalds, and then by many others
- Minix runs on PC architectures, should be small enough to be entirely understandable within one course (except hard disk driver)
- Minix is well documented in our textbook

Variety of Operating Systems

- embedded system, e.g. for a sensor controller, must execute tasks in a certain sequence and at certain times
- embedded system, e.g. for a car, may have to react to commands from the user and to sensor inputs, e.g. so as to control airbags
- server, e.g. for a web page or for a collection of virtual machines, may have to assign resources so the services are performed at a minimum cost in hardware, and must provide security
- user workstation, must provide security and assign resources so user gets quick response, and also must be very flexible (general purpose)
- compute engine may have to manage multiple tightly coupled processors with shared memory so as many computations as possible complete as soon as possible
- distributed operating system has to do all of the above, but also provide redundancy and robustness in case of failure of one or more subsystems
- grid computing system has to deal with widespread fluctuation in processor availability and capability, and may have to protect against computers that perform malicious operations

Your instructor's OS background

- mostly user, some design work
- Multics, 1978
- Unix, 1979, 1985-201x (including SunOS, Solaris)
- some DEC real-time system, 1980-1982
- Data General operating system, 1982
- MS-DOS and windows, 1982-
- VAX VMS, 1982-1985
- Apple II operating system and P-machine, 1982-1985
- Medos-2, 1982-1985
- custom-built computer, 1982-1985
- Apple Lisa and Macintosh, 1980s
- operating system concepts for experimental research machine (simulated but never built), 1985-1991
- Xinu, 1986?
- operating system for the MasPar 1024-node machine, 1989-1991
- control software for a network switch, 1992-1994
- Mach, 1993-1997
- Foxnet network stack with some operating system functions, 1993-1997
- Linux, 1994-
- Hello operating system, 1998-1999
- Minix, 2004-2018
- TinyOS, 2001-2004

Changes from the 2011 Course

- seL4: an operating system with a proof of correctness
- more Linux

Thursday January 10th

- bring your computing environment to class
- install Minix on one virtual machine (or partition)
- if time allows: install Linux on a different virtual machine (or partition)
- if time allows: install seL4 on a different virtual machine