Operating Systems Design and Implementation COMPSCI 577

Prashant Shenoy

University of Massachusetts Amherst

Today's Class

- Organizational meeting
  - Course organization & outline
  - Policies
  - Prerequisites & course sign-up
- Introduction

Organizational Information

- Course Resources
  - **Web** http://lass.cs.umass.edu/~shenoy/courses/577
  - **Piazza** discussion forms
  - **Github Classroom** for programming assignments
  - **Moodle** for online grade book
  - **YouTube** for recorded lectures
- Contact info: shenoy [at] cs.umass.edu
- Course Staff
  - Secondary Instructor: Dr. Ahmed Ali-Eldin
    - ahmeda [at] cs.umass.edu
  - **TA:** Walid Hanafy
    - whanafy [at] cs.umass.edu

Course Staff

Prashant Shenoy (instructor)

Ahmed Ali-Eldin (2nd instructor)

Walid Hanafy - TA

Prerequisites

- Undergrad Operating Systems (COMPSCI 377)
  - Review: OS Textbook: Operating Systems: Three Easy Pieces (Available for free online)
  - 377 Lectures: youtube.com/umassos
    - Click Playlist and choose 377 (S16)
- C Programming
  - All operating systems are written in C, and our assignments / project will also be in C
  - Review: C Programming Language, Kerninghan and Ritchie & C Programming for Absolute Beginners

Course Requirements

- Course grading
  - 5-6 programming assignments (50%)
  - Final Project (25%)
  - One midterm (20%)
  - Class participation (5%)
- Textbook: No required textbook
  - Suggested reference: Operating System Design and Implementation, 3rd ed, Tannenbaum and Woodhull (``Minix book'')
  - Course materials will be made available online

Course Organization: Misc

- Personal laptop or desktop for programming
- Ed-lab access: 30+ Linux-based PCs
- Office hours:
  - Instructor: Thu: 1:30-2:30, LGRC 333 or by appt
  - TA Office hrs and location: to be announced

Assignments and Projects

- 5-6 ``short'' assignments
  - Class lectures will provide background
- One Final Project
- Assignments/projects will use C and build on existing operating systems

- Github Classroom for all programming
- Provide your github ID to TA via Moodle questionaire #1.

Plagiarism

- Cheating includes:
  - "Borrowing" code from someone
    - This includes reading previous solutions
  - Giving code to someone (even next year)
  - Copying code from anyone (including the net)
  - Hiring someone to write your code
  - Submitting someone else's code as your own
  - Looking at anyone else's code

Cell Phone and Laptop Policy

- Cell phones should be off or on silent alert
- Texting is strictly prohibited in class
- Laptops and tablets may **NOT** be used during lectures: No email, browsing, facebook, twitter during class lectures
- Laptops **allowed** during in-class programming work
- Penalty of 2 points per violation, plus other penalties

What is this course about?

- Advanced course on Operating Systems
- Focuses on OS Internals
  - UNIX-like OS (Minix & Linux)
- Hand-on exposure to OS
  - learn by doing, "design and implement"

Objectives & Learning Outcomes

- **http://lass.cs.umass.edu/~shenoy/courses/577/outcomes.html**
- **Learning Outcomes**
  - Developing low-level operating system code.
  - Understanding the performance and design trade-offs in complex software systems
  - Understanding and be capable of developing OS code inside a variety of OS environments, including monolithic, microkernels, and virtual machines, including device drivers.
  - Developing benchmarks and use of profiling tools to evaluate the performance of operating systems and application stacks.
  - Understanding and of evaluating research published in the field of operating systems

Designing Large Systems

- OS as an example of large system design
- Goals: Fast, reliable, large scale
- To build these systems, you need to know
  - Each computer:
    - Architectural details that matter
    - C and C++ (nitty gritty & more)
    - Memory management & locality
    - Concurrency & scheduling
    - Disks, network, file systems
  - Across cluster:
    - Server architectures
    - Distributed computing, file systems

Course Outline & Topics

- OS Architecture, Micro-kernels
- Processes Management
- Memory Management
- I/O, Storage and File Systems
- Virtualization and Cluster Scheduling

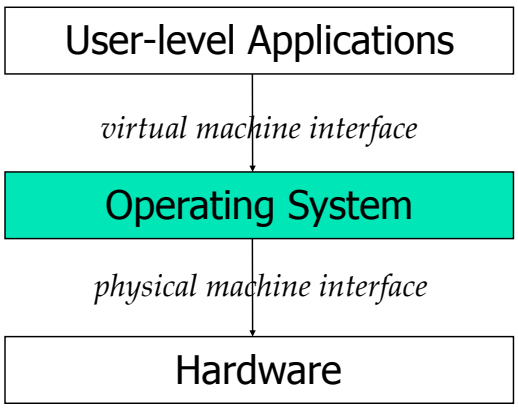
- 577: Sits between “OS concepts (377)” and “distributed systems (677)

What's An Operating System?

- Definition has changed over years
  - Originally, very bare bones
  - Now, includes more and more
- Operating System (OS)
  - Interface between the user and the architecture
  - Implements a virtual machine that is
  - (hopefully) easier to program than raw hardware.

OS: Traditional View

- Interface between user and architecture
  - Hides architectural details
- Implements virtual machine:
  - Easier to program than raw hardware
- Illusionist
  - Bigger, faster, reliable
- Government
  - Divides resources
  - "Taxes" = overhead

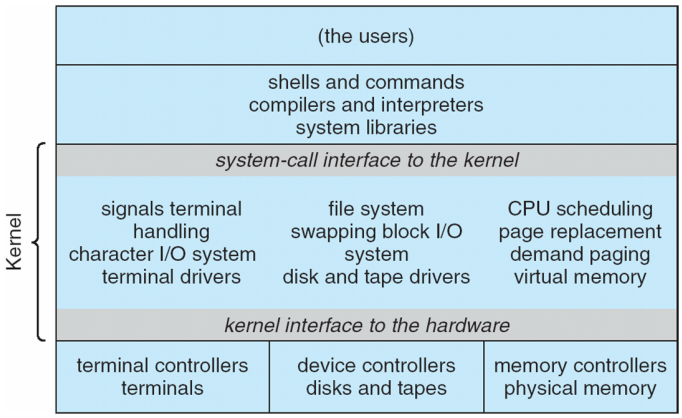


New Developments in OS

- Operating systems: active field of research
  - Demands on OS's growing
  - New application spaces (Web, Grid, Cloud)
  - Rapidly evolving hardware

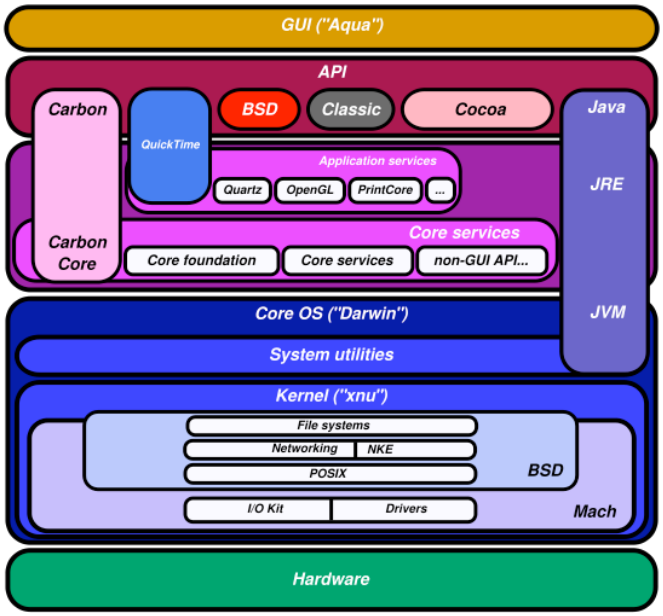
- Advent of open-source operating systems
  - Linux etc.
  - You can contribute to and develop OS's!
  - Excellent research platform

One Basic OS Structure

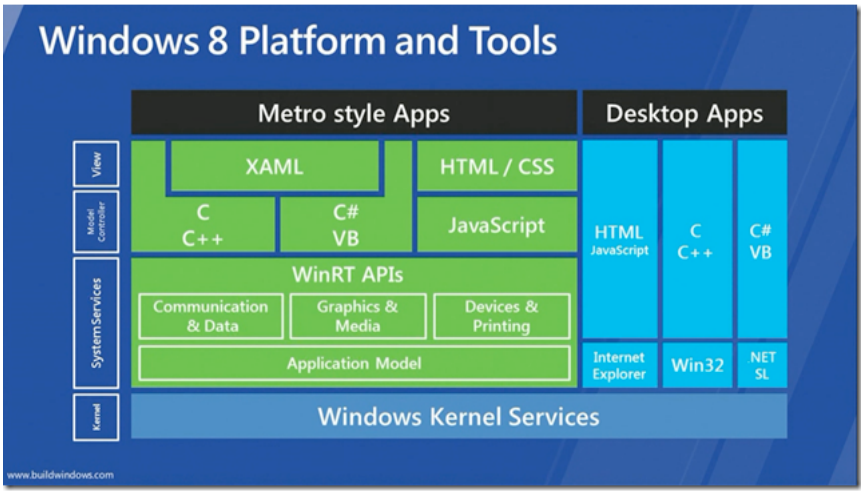


- The *kernel* is the protected part of the OS that runs in kernel mode, protecting the critical OS data structures and device registers from user programs.
- Debate about what functionality goes into the kernel (above figure: UNIX) - "**monolithic** kernels"

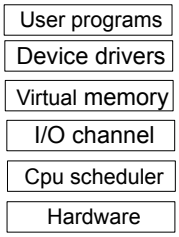
Mac OS X Architecture



Windows Architecture



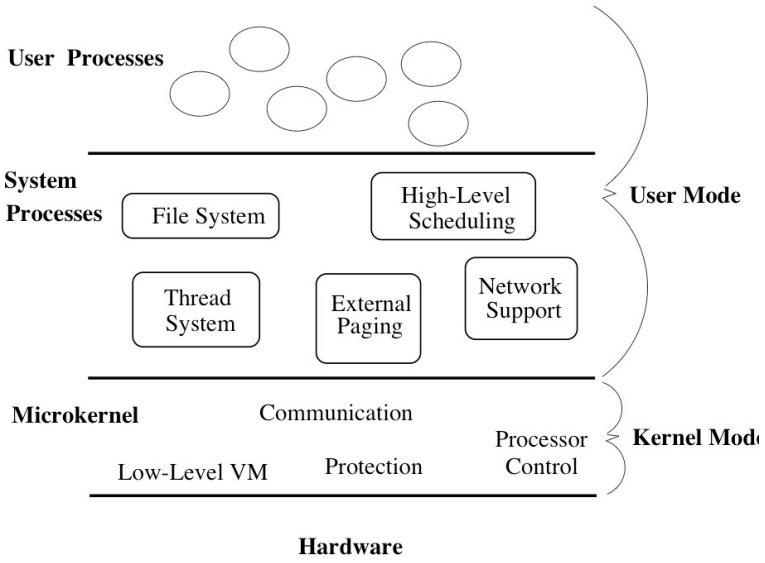
Layered OS design



Layer N: uses layer N-1 and provides new functionality to N+1

- Advantages: modularity, simplicity, portability, ease of design/debugging
- Disadvantage - communication overhead between layers, extra copying, book-keeping

Microkernel

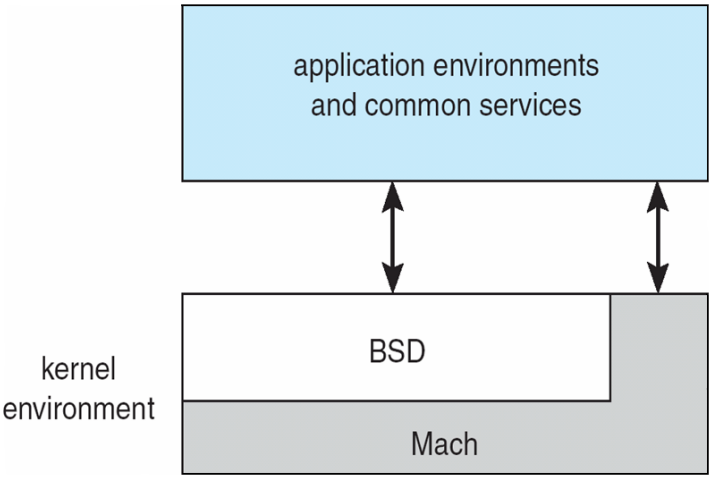


- Small kernel that provides communication (message passing) and other basic functionality
  - other OS functionality implemented as user-space processes

Microkernel Features

- **Goal**: to minimize what goes in the kernel (mechanism, no policy), implementing as much of the OS in User-Level processes as possible.
- Advantages
  - better reliability, easier extension and customization
  - mediocre performance (unfortunately)
- First Microkernel was Hydra (CMU '70). Current systems include Chorus (France) and Mach (CMU).
- We will be using Minix in this course

Mac OS X - hybrid approach



- Layered system: Mach microkernel (mem, RPC, IPC) + BSD (threads, CLI, networking, filesystem) + user-level services (GUI)

Modules

- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible

*NIX Modular Approach