Welcome to Operating Systems!

Operating system: the single-most complex and essential software you run on your machine

In this class, we will explore how an OS works

• Basic concepts
• Structure, design, implementation
• Principles that apply to all OS’s

This is my favorite class! I hope it will be yours too!
Introductions

Instructor

• Prof. Joe Pasquale
• APM 5121

TA’s

• Jesse Steinberg
• John Ehrhard
• Sriram Ramabhadran
• Sunny Chow
Resources

Web page

- http://www-cse.ucsd.edu/classes/fa03/cse120/

Lecture notes

- Will be posted day before lecture

Book

- Operating System Concepts, 6th Edition
  Silberschatz, Galvin, Gagne
  Wiley, 2003
Lectures vs. Book

Lectures are very important: Don’t miss them!

Designed to highlight what is most important to know

Exam questions will come directly from lectures
  • Lecture notes + what is said in class

Use the book to fill in details and gaps
Grading

30% Midterm

40% Final

30% Programming Assignments (3)
  • Can collaborate, but must submit your own work
  • Exams will include questions on programming

Collaborate: discuss problems, approaches, not solutions
How to Ace this Class

Getting the most out of lectures

• *Come to class with lecture notes, annotate*
• Afterwards, read book using notes as a guide
• Prepare by reading book before class

Preparing for exams

• Study the notes carefully
• *Using notes as your guide, study book*

Programming assignments: get an *early start!*
What is an Operating System?

Basically, software that enhances the hardware

But what does it do? How does it help?

• Helps you (as a user or programmer) by making the system easier to use
• Helps your programs run by providing resources and protecting them
• Helps the system by keeping things running smoothly
We Focus on the Kernel

The kernel:

• All programs depend on it to run
• Operates closely with the hardware
• Allocates basic resources like the CPU and memory
• Controls I/O devices

When we say “OS” we mean the kernel
Two Goals of Operating Systems

1. Manage/coordinate hardware resources so that the system operates smoothly: efficiently, reliably, securely

2. Present abstract system model to programmer that promotes simple and convenient access to and control of resources
Manager/Coordinator of Resources

Coordinates who gets what

- “who”: running programs
- “what”: resources
- “when”: scheduling time
- “where”: organizing space
- “whether”: limits, rights

Goal: smooth system operation

- efficiency, reliability, security
Present Abstract System Model

OS provides abstractions for resources, how they operate and interact, and policies to manage them

- “who”: processes
- “what”: contexts, segments, files, sockets, etc.
- “when”: deadlines, priority, round-robin, etc.
- “where”: best-fit, first-fit, contiguous, etc.
- “whether”: exclusive-access, read-only, etc.

Goal: simple/convenient access/control of resources
What If No Operating System?

All we have is bare hardware

You want to run a program

• How do you load it?
• How do you run it?
• What happens when it completes?

Need at least some minimal OS to do these functions

Bare Hardware
User Program
Provides Ability to Run a Program

Minimal kernel

- resident code that runs by default
- allows you to load memory with program and run
- when done return to kernel

Questions

- What happens if program fails or has a bug?
- How is kernel protected?
Provides Commonly Needed Functions

Some functions needed by many programs

- I/O device control, memory allocation, etc.
- Place these functions in kernel, called by programs

What should functions be?

- How many programs should benefit?
- Might kernel get too big?
Allowing Multiple Programs to Run

When I/O request is made, CPU becomes idle

- allow another program to run: multiprogramming
- requires yielding (giving up CPU) and sharing memory

What if one running program

- monopolizes CPU, memory?
- reads/writes another’s memory?
- uses I/O device being used by another?
Creating Illusions

Multiple virtual processors by rapidly switching use of CPU
Multiple virtual memories by memory partitioning and re-addressing
Idealized devices by simplifying interfaces, and using other resources to enhance function
Bottom line: make the system easy to use and work well
Outline of Course

- Processes
- Virtual Memory
- I/O System
- File System
- Protection and Security
- Distributed Systems and Networks