Welcome to Operating Systems!

Operating system: the single-most complex and essential software running 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 (pasquale@cs.ucsd.edu)
• EBU3B 3112

TA’s

• John Fisher-Ogden (jfisherogden@cs.ucsd.edu)
• AfmZakaria Haque (ahaque@cs.ucsd.edu)

Discussion Section

• Mondays, 10:00-10:50, Center 212
Resources

Web page
- http://www-cse.ucsd.edu/classes/wi06/cse120/

Lecture notes
- Available via web page evening before lecture

Book

Webboard
- http://webboard.ucsd.edu/WB/?boardid=cs120w

Computer system (for programming assignments)
- ieng9.ucsd.edu
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 as a reference, to fill in details and gaps
Grading

25% Midterm exam

40% Final exam

10% Homework exercises

25% Programming projects
Collaboration Policy

Can collaborate, but must submit your own work

Exams will include questions on homework, programming

Collaborate: discuss approaches, not solutions

Test: Can you reproduce and explain it, all by yourself?
How to Ace this Class

Getting the most out of lectures
• Prepare by reading book before class
• *Come to class* with lecture notes, annotate
• Afterwards, read book using notes as a guide

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

Do all the homework

Programming projects: get an early start
What is an Operating System?

Basically, software the enhances the hardware

- Provides *interface* so that system is *easier to use*
- Provides *resources* to *allow programs to run*
- *Protects* resources and running programs
- Keeps the system *running smoothly*

So why not just do everything in hardware?
Some Key Terms

Hardware
- All the physical working parts

Resources
- What are needed to allow work to get done

Operating System
- Software that enhances the hardware

Kernel
- The essential part ("core") of the operating system

"The System"
- Generally all of the above, viewed in a unified way
In this Class, We Focus on the Kernel

All programs depend on it
- Loads and runs them
- Accessed via system calls

Works closely with hardware
- Access device registers
- Responds to interrupts

Allocates basic resources
- CPU time, memory space

Controls I/O devices
- Display, keyboard, disk, network, ...

<table>
<thead>
<tr>
<th>User Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
</tbody>
</table>
Two Purposes of Operating System

Provides abstract machine
- Functions and resources
- Goals
  - simplicity, convenience

Manages resources
- Allocates space and time
- Goals
  - efficiency, reliability
  - protection, security
Resources and Abstractions

Hardware Abstraction

CPU process, thread
Memory segment, page
Disk file, directory
Network message, port
Display window
Keyboard stream

Resource: something that allows work to get done

Abstraction: a simplified representation or model
What If There Is No Kernel?

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 exits?

Need at least a minimal kernel to do these functions
Minimal Kernel: Allow Program To Run

Minimal kernel

• Resident code
• Runs by default
• Loads program into memory
• Allows it to run
• When it exits, go to kernel

Questions

• What if program fails or has a bug?
• How is kernel protected?
Provide Common Functions

Some functions are useful to many programs
  • I/O device control
  • Memory allocation

Place these functions in kernel
  • Called by programs
  • Or accessed implicitly

What should functions be?
  • How many programs should benefit?
  • Might kernel get too big?
Allow Multiple Programs to Run

When I/O issued, CPU not needed

- 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?

<table>
<thead>
<tr>
<th>User Programs</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>CPU</td>
</tr>
<tr>
<td>P2</td>
<td>Memory</td>
</tr>
<tr>
<td></td>
<td>I/O Dev ... I/O Dev</td>
</tr>
</tbody>
</table>
Virtualize, Idealize (Abstract)

Multiple virtual processors
- by rapidly switching CPU use

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
• File System
• I/O
• Protection and Security
• Distributed Systems and Networks
Reading and Homework

Read Chapters 1 and 2

• **Review hardware material**
  • You are expected to already know this
  • If not, you may need to do further research

• **Get familiar with operating system concepts**
  • Just get to know terms, ideas
  • Later, this material will be good to refer to

Note first homework assignment (on web page)

• Due Friday 1/13 at 5pm (corrected)