

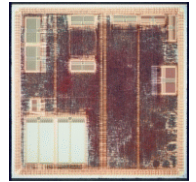
# CS162 Operating Systems and Systems Programming Lecture 1

## What is an Operating System?

August 29, 2005  
Prof. John Kubiatowicz  
<http://inst.eecs.berkeley.edu/~cs162>

## Who am I?

- Professor John Kubiatowicz (Prof "Kubi")
  - Background in Hardware Design
    - » Alewife project at MIT
    - » Designed CMMU, Modified SPARC processor
    - » Helped to write operating system
  - Background in Operating Systems
    - » Worked for Project Athena (MIT)
    - » OS Developer (device drivers, network file systems)
    - » Worked on Clustered High-Availability systems (CLAM Associates)
  - Peer-to-Peer
    - » OceanStore project - Store your data for 1000 years
    - » Tapestry and Bamboo - Find you data around globe
  - Quantum Computing
    - » Well, this is just cool, but probably not apropos



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.2

## Goals for Today

- What is an Operating System?
  - And - what is it not?
- Examples of Operating Systems design
- Why study Operating Systems?
- Oh, and "How does this class operate?"

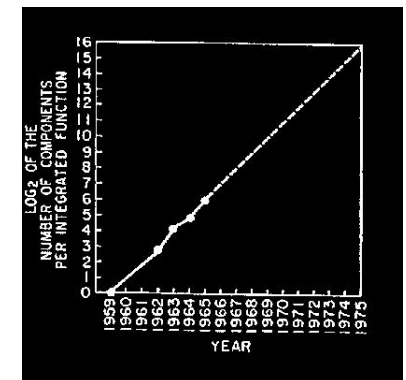
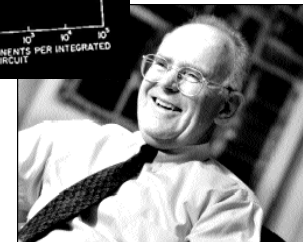
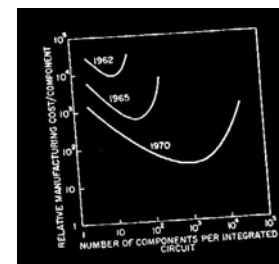
Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.3

## Rapid Underlying Technology Change



- "Cramming More Components onto Integrated Circuits"
  - Gordon Moore, Electronics, 1965

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.4

## Computing Devices Everywhere



8/29/05

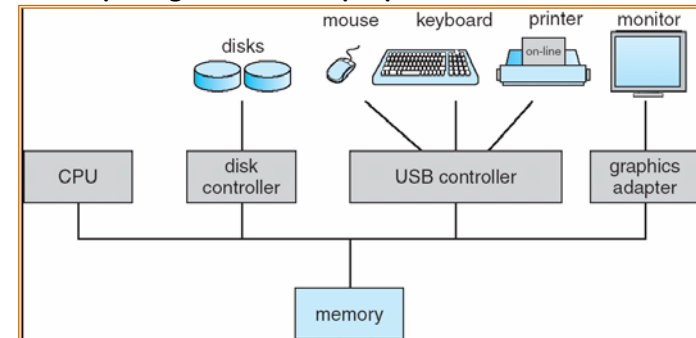
Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.5

## Computer System Organization

### • Computer-system operation

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles

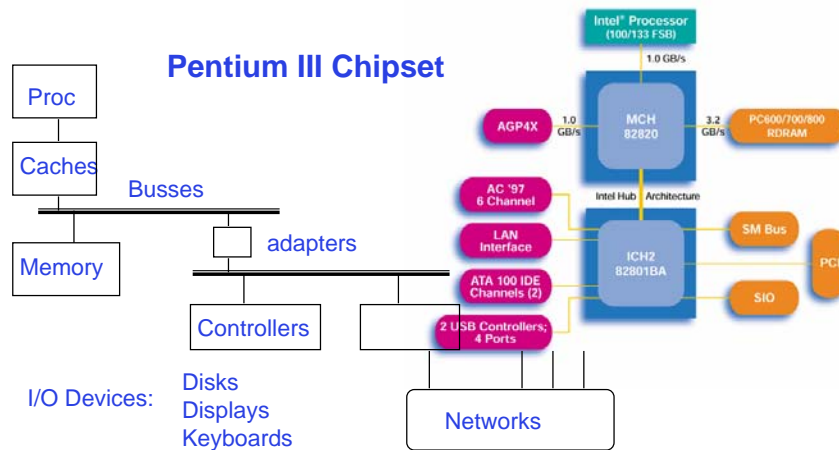


8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.6

## Functionality comes with great complexity!



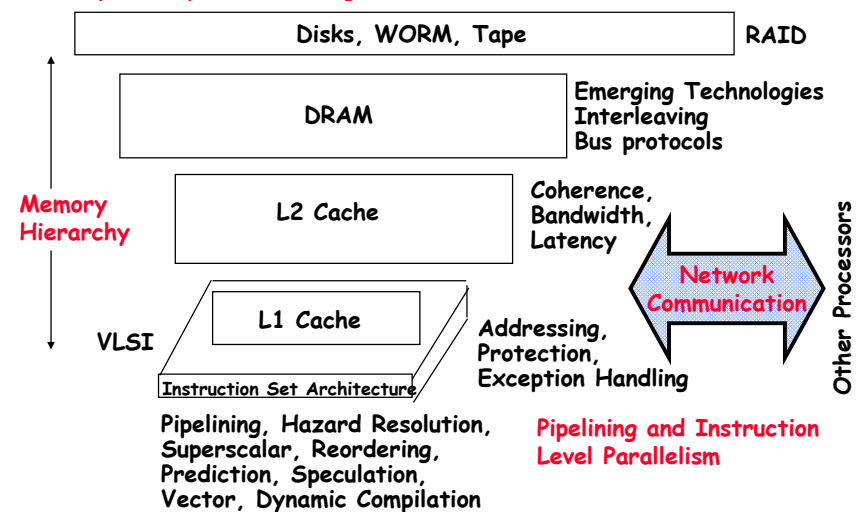
8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.7

## Sample of Computer Architecture Topics

### Input/Output and Storage



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.8

## Example: Some Mars Rover Requirements

- Serious hardware limitations/complexity:
  - 20Mhz powerPC processor, 128MB of RAM
  - cameras, scientific instruments, batteries, solar panels, and locomotion equipment
  - Many independent processes work together
- Can't hit reset button very easily!
  - Must reboot itself if necessary
  - Always able to receive commands from Earth
- Individual Programs must not interfere
  - Suppose the MUT (Martian Universal Translator Module) buggy
  - Better not crash antenna positioning software!
- Further, all software may crash occasionally
  - Automatic restart with diagnostics sent to Earth
  - Periodic checkpoint of results saved?
- Certain functions time critical:
  - Need to stop before hitting something
  - Must track orbit of Earth for communication



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.9

## How do we tame complexity?

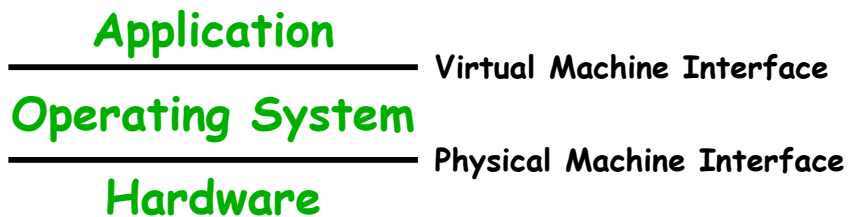
- Every piece of computer hardware different
  - Different CPU
    - » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, ...
  - Different types of devices
    - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - » Cable, DSL, Wireless, Firewalls,...
- Questions:
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.10

## OS Tool: Virtual Machine Abstraction



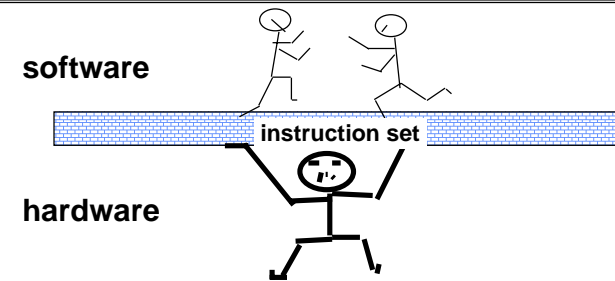
- Software Engineering Problem:
  - Turn hardware/software quirks ⇒ what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc...
- For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What's the hardware interface? (physical reality)
  - What's the application interface? (nicer abstraction)

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.11

## Interfaces Provide Important Boundaries



- Why do interfaces look the way that they do?
  - History, Functionality, Stupidity, Bugs, Management
  - CS152 ⇒ Machine interface
  - CS160 ⇒ Human interface
- Should responsibilities be pushed across boundaries?
  - RISC architectures
  - Graphical Interfaces

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.12

## Virtual Machines

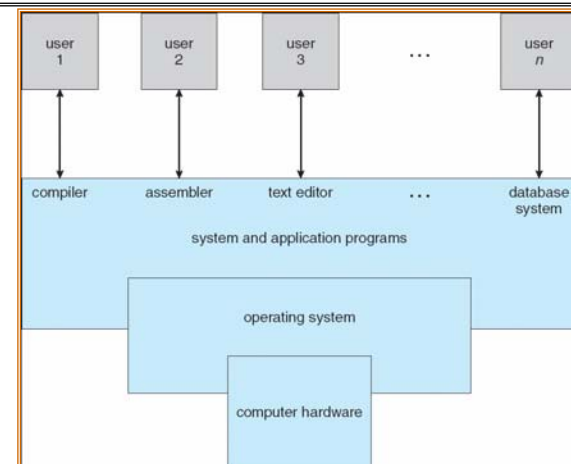
- Software emulation of an abstract machine
  - Make it look like hardware has features you want
  - Programs from one hardware & OS on another one
- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different Devices appear to have same interface
  - Device Interfaces more powerful than raw hardware
    - » Bitmapped display ⇒ windowing system
    - » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)
- Fault Isolation
  - Processes unable to directly impact other processes
  - Bugs cannot crash whole machine
- Protection and Portability
  - Java interface safe and stable across many platforms

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.13

## Four Components of a Computer System



**Definition:** An operating system implements a virtual machine that is (hopefully) easier and safer to program and use than the raw hardware.

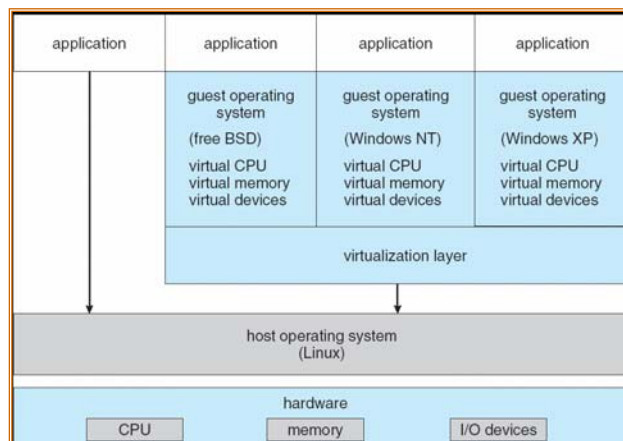
8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.14

## Virtual Machines (con't): Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

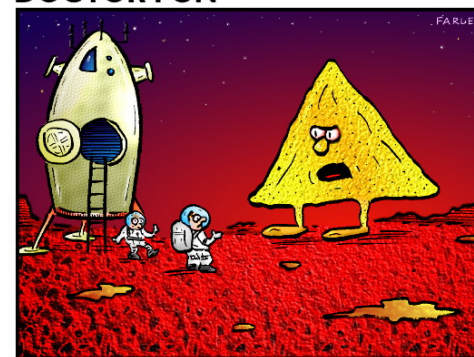
Lec 1.15

## Nachos: Virtual OS Environment

- You will be working with Nachos
  - Simulation environment
  - Hardware, interrupts, I/O
  - Execution of User Programs running on this platform

### DOCTOR FUN

6 Dec 94



© Copyright 1994 David Farney. World rights reserved. All rights reserved. This illustration is for personal viewing only. All other rights reserved. Opinions expressed herein are not those of the University of Chicago or the University of North Carolina.

"This is the planet where nachos rule."

8/29/05

Lec 1.16

## What does an Operating System do?

---

- Silerschatz and Gavin:  
"An OS is Similar to a government"
  - Begg the question: does a government do anything useful by itself?
- Coordinator and Traffic Cop:
  - Manages all resources
  - Settles conflicting requests for resources
  - Prevent errors and improper use of the computer
- Facilitator:
  - Provides facilities that everyone needs
  - Standard Libraries, Windowing systems
  - Make application programming easier, faster, less error-prone
- Some features reflect both tasks:
  - E.g. File system is needed by everyone (Facilitator)
  - But File system must be Protected (Traffic Cop)

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.17

## What is an Operating System,... Really?

---

- Most Likely:
  - Memory Management
  - I/O Management
  - CPU Scheduling
  - Communications? (Does Email belong in OS?)
  - Multitasking/multiprogramming?
- What about?
  - File System?
  - Multimedia Support?
  - User Interface?
  - Internet Browser? ☺
- Is this only interesting to Academics??

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.18

## Operating System Definition (Cont.)

---

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
  - Everything else is either a system program (ships with the operating system) or an application program

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.19

## Course Administration

---

- Instructor: John Kubiatowicz ([kubitron@cs.berkeley.edu](mailto:kubitron@cs.berkeley.edu))  
673 Soda Hall  
Office Hours(Tentative): M/W 2:00-3:00pm
- TAs: Dominic Antonelli ([cs162-ta@cory](mailto:cs162-ta@cory))  
Christopher Louis Portka ([cs162-tb@cory](mailto:cs162-tb@cory))  
Rajesh Nishtala ([cs162-tc@cory](mailto:cs162-tc@cory))
- Labs: Second floor of Soda Hall
- Materials: <http://inst.eecs.berkeley.edu/~cs162>
- Mirror: <http://www.cs.berkeley.edu/~kubitron/cs162>
- Newsgroup: [ucb.class.cs162](mailto:ucb.class.cs162)
- TA Email: [cs162@cory.cs.berkeley.edu](mailto:cs162@cory.cs.berkeley.edu)
- Are you on the waitlist?
  - Need to speak with Michael-David in 379 Soda Hall
  - Fill out blue appeal form

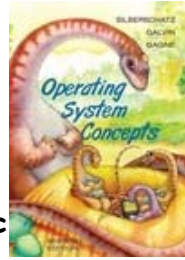
8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.20

## Textbook

- Text: *Operating Systems Concepts*, 7<sup>th</sup> Edition Silberschatz, Galvin, Gagne
- Online supplements
  - See "Information" link on course website
  - Includes Appendices, sample problems, etc
- Question: need 7<sup>th</sup> edition?
  - No, but has new material that we may cover
  - Completely reorganized
  - Will try to give readings from both the 6<sup>th</sup> and 7<sup>th</sup> editions on the lecture page



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.21

## Topic Coverage

**Textbook: Silberschatz, Galvin, and Gagne, *Operating Systems Concepts*, 7<sup>th</sup> Ed., 2005**

- 1 week: Fundamentals (Operating Systems Structures)
- 1.5 weeks: Process Control and Threads
- 2.5 weeks: Synchronization and scheduling
- 2 week: Protection, Address translation, Caching
- 1 week: Demand Paging
- 1 week: File Systems
- 2.5 weeks: Networking and Distributed Systems
- 1 week: Protection and Security
- ???: Advanced topics: Peer-to-peer, Smart Dust, others?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.22

## Class Schedule

- Class Time: M/W 5:30 - 7:00pm, 10 Evans
  - Awake! Awake! Awake!
  - Please come to class. Lecture notes do not have everything in them. The best part of class is the interaction!
- Sections:
  - Important information is in the sections
  - The sections assigned to you by Telebears are temporary!
  - Every member of group must be in same section

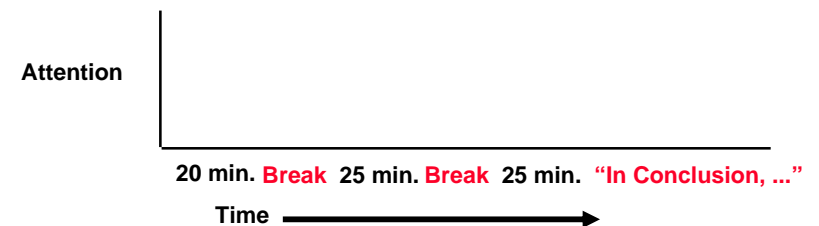
Section	Time	Location	TA
101	Tu 1:00-2:00P	310 Hearst Mining	Dominic
102	W 10:00-11:00A	2 Evans	Rajesh
103	W 11:00-12:00P	85 Evans	Rajesh
104	W 1:00-2:00P	85 Evans	Chris
105	W 2:00-3:00P	85 Evans	Chris

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.23

## Typical Lecture Format



- 1-Minute Review
- 20-Minute Lecture
- 5-Minute Administrative Matters
- 25-Minute Lecture
- 5-Minute Break (water, stretch)
- 25-Minute Lecture
- Instructor will come to class early & stay after to answer questions

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.24

## Lecture Goal

---

**Interactive!!!**

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.25

## Computing Facilities

---

- Every student who is enrolled should get an account form at end of lecture
  - Gives you an account of form cs162-xx@cory
  - This account is required
    - » Most of your debugging can be done on other EECS accounts, however...
    - » All of the final runs must be done on your cs162-xx account and must run on the x86 Solaris machines
- Make sure to log into your new account this week and fill out the questions
- Project Information:
  - See the "Projects and Nachos" link off the course home page
- Newsgroup (ucb.class.cs162):
  - Read this regularly!

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.26

## Grading

---

- Rough Grade Breakdown
  - Two Midterms: 15% each
  - One Final: 15%
  - Four Projects: 50% (i.e. 12.5% each)
  - Participation: 5%
- Four Projects:
  - Phase I: Build a thread system
  - Phase II: Implement Multithreading
  - Phase III: Caching and Virtual Memory
  - Phase IV: Networking and Distributed Systems
- Late Policy:
  - Each student has 5 "slip" days.
  - For Projects, slip days deducted from all partners
  - 10% off per day after slip days exhausted

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.27

## Academic Dishonesty Policy

---

- Copying all or part of another person's work, or using reference material not specifically allowed, are forms of cheating and will not be tolerated. A student involved in an incident of cheating will be notified by the instructor and the following policy will apply:  
<http://www.eecs.berkeley.edu/Policies/acad.dis.shtml>
- The instructor may take actions such as:
  - require repetition of the subject work,
  - assign an F grade or a 'zero' grade to the subject work,
  - for serious offenses, assign an F grade for the course.
- The instructor must inform the student and the Department Chair in writing of the incident, the action taken, if any, and the student's right to appeal to the Chair of the Department Grievance Committee or to the Director of the Office of Student Conduct.
- The Office of Student Conduct may choose to conduct a formal hearing on the incident and to assess a penalty for misconduct.
- The Department will recommend that students involved in a second incident of cheating be dismissed from the University.

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.28

## Group Project Simulates Industrial Environment

- Project teams have 4 or 5 members in same discussion section
  - Must work in groups in "the real world"
- Communicate with colleagues (team members)
  - Communication problems are natural
  - What have you done?
  - What answers you need from others?
  - You must document your work!!!
  - Everyone must keep an on-line notebook
- Communicate with supervisor (TAs)
  - How is the team's plan?
  - Short progress reports are required:
    - » What is the team's game plan?
    - » What is each member's responsibility?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.29

## What if we didn't have an Operating System?

- Source Code⇒Compiler⇒Object Code⇒Hardware
- How do you get object code onto the hardware?
- How do you print out the answer?
- Once upon a time, had to Toggle in program in binary and read out answer from LED's!



**Altair 8080**

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.30

## Simple OS: What if only one application?

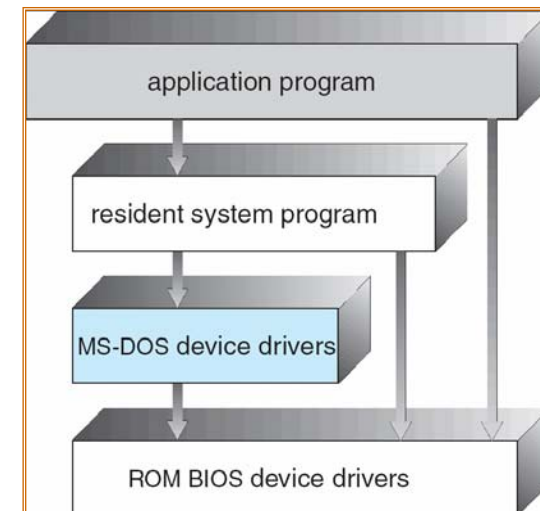
- Examples:
  - Very early computers
  - Early PCs
  - Embedded controllers (elevators, cars, etc)
- OS becomes just a library of standard services
  - Standard device drivers
  - Interrupt handlers
  - Math libraries

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.31

## MS-DOS Layer Structure



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.32



## More thoughts on Simple OS

- What about Cell-phones, Xboxes, etc?
  - Is this organization enough?
- Can OS be encoded in ROM/Flash ROM?
- Does OS have to be software?
  - Can it be Hardware?
  - Custom Chip with predefined behavior
  - Are these even OSs?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.33

## More complex OS: Multiple Apps

- Full Coordination and Protection
  - Manage interactions between different users
  - Multiple programs running simultaneously
  - Multiplex and protect Hardware Resources
    - » CPU, Memory, I/O devices like disks, printers, etc
- Facilitator
  - Still provides Standard libraries, facilities
- Would this complexity make sense if there were only one application that you cared about?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.34

## Example: Protecting Processes from Each Other

- Problem: Run multiple applications in such a way that they are protected from one another
- Goal:
  - Keep User Programs from Crashing OS
  - Keep User Programs from Crashing each other
  - [Keep Parts of OS from crashing other parts?]
- (Some of the required) Mechanisms:
  - Address Translation
  - Dual Mode Operation
- Simple Policy:
  - Programs are not allowed to read/write memory of other Programs or of Operating System

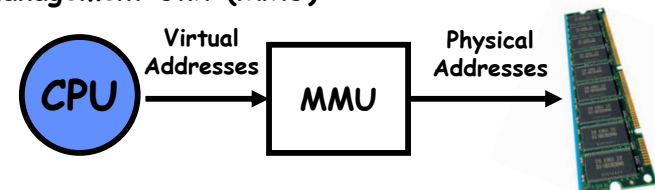
8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.35

## Address Translation

- Address Space
  - A group of memory addresses usable by something
  - Each program (process) and kernel has potentially different address spaces.
- Address Translation:
  - Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  - Mapping *often* performed in Hardware by Memory Management Unit (MMU)

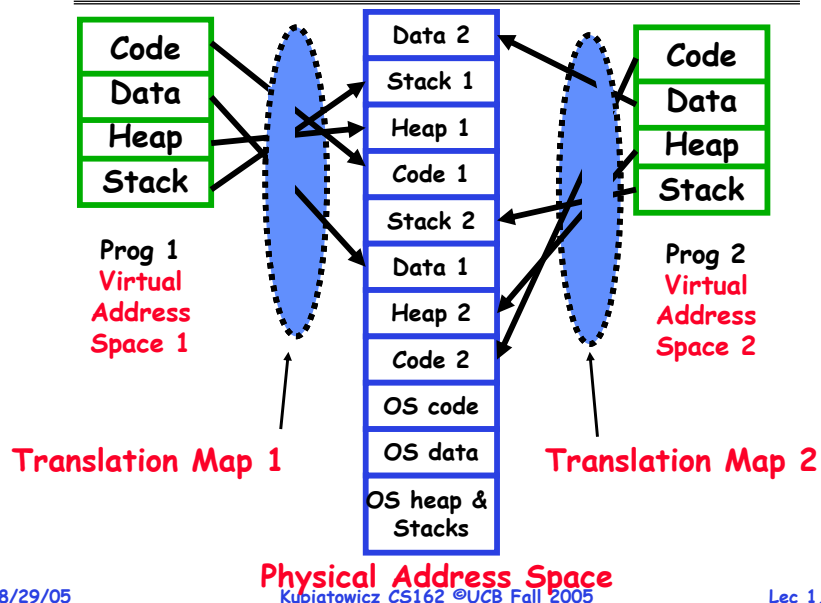


8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.36

## Example of Address Translation



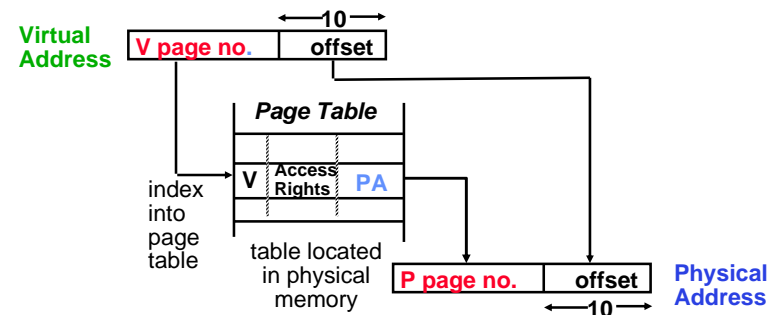
8/29/05

Physical Address Space  
Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.37

## Address Translation Details

- For now, assume translation happens with table (called a Page Table):



- Translation helps protection:
  - Control translations, control access
  - Should Users be able to change Page Table???

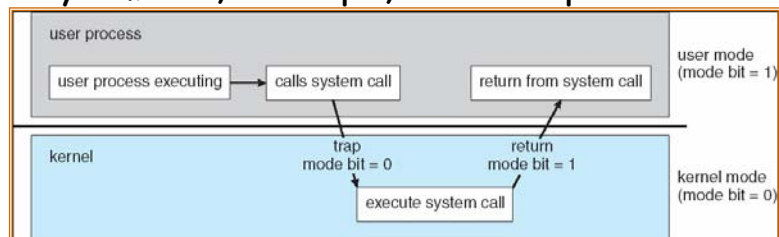
8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.38

## Dual Mode Operation

- Hardware provides at least two modes:
  - "Kernel" mode (or "supervisor" or "protected")
  - "User" mode: Normal programs executed
- Some instructions/ops prohibited in user mode:
  - Example: cannot modify page tables in user mode
    - » Attempt to modify ⇒ Exception generated
- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions

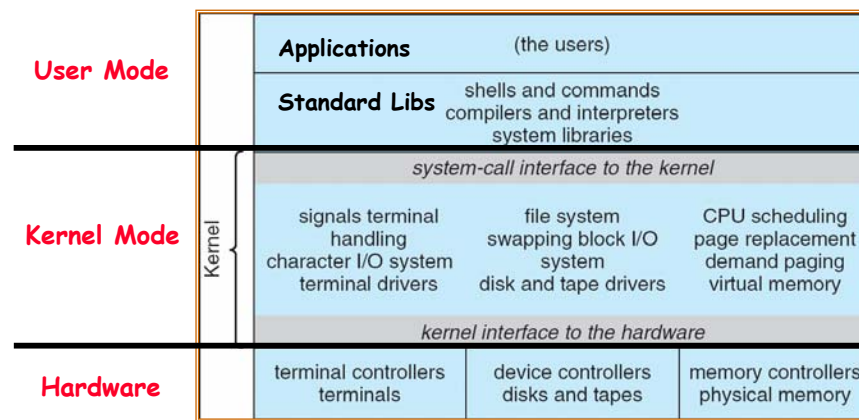


8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.39

## UNIX System Structure



8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.40

## OS Systems Principles

---

- OS as illusionist:
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- OS as government:
  - Protect users from each other
  - Allocate resources efficiently and fairly
- OS as complex system:
  - Constant tension between simplicity and functionality or performance
- OS as history teacher
  - Learn from past
  - Adapt as hardware tradeoffs change

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.41

## Why Study Operating Systems?

---

- Learn how to build complex systems:
  - How can you manage complexity for future projects?
- Engineering issues:
  - Why is the web so slow sometimes? Can you fix it?
  - What features should be in the next mars Rover?
  - How do large distributed systems work? (Kazaa, etc)
- Buying and using a personal computer:
  - Why different PCs with same CPU behave differently
  - How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? [ Ok, made last one up ]
  - Should you get Windows XP, 2000, Linux, Mac OS ...?
  - Why does Microsoft have such a bad name?
- Business issues:
  - Should your division buy thin-clients vs PC?
- Security, viruses, and worms
  - What exposure do you have to worry about?

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.42

## "In conclusion..."

---

- Operating systems provide a virtual machine abstraction to handle diverse hardware
- Operating systems coordinate resources and protect users from each other
- Operating systems simplify application development by providing standard services
- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery
- CS162 combines things from many other areas of computer science -
  - Languages, data structures, hardware, and algorithms

8/29/05

Kubiatowicz CS162 ©UCB Fall 2005

Lec 1.43