Introduction

John Criswell
University of Rochester

Logistics

Course Instructors

Instructor	Name: John Criswell Email: criswell@cs Office: CSB 717 Office Hours: 4:00 - 5:00 Wednesdays
TA	Name: To Be Announced Email: To Be Announced Office: To Be Announced Office Hours: To Be Announced

About Me

- Research Interests:
  - Computer Security
  - Operating Systems
    - Compilers
- Trusted Operating Systems: Solaris and AIX
- Protect OS kernel from memory safety attacks
- Protect applications from OS kernels

Prerequisites

- CSC 252 or equivalent
- C/C++ programming experience on Unix systems

Recommended Textbooks

- Concepts
  - *Modern Operating Systems* by Andrew Tanenbaum
  - *Operating System Concepts* by Silberschatz et al.
- Linux Kernel Internals
  - *Understanding the Linux Kernel* by Bovet and Cesati
  - *Professional Linux Kernel Architecture* by Mauerer
  - *Linux Kernel Development* by Love
- Unix API
  - *Advanced Programming in the UNIX Environment* by Stevens and Rago

Additional Resources

- Books available online via the library
  - *The Design and Implementation of the FreeBSD Operating System*
- Papers available on the Internet
  - Links will be provided on course web page or Blackboard

Grading for Undergraduates

- Five (5) programming assignments: 50%
- Five (5) written homework assignments: 10%
- Midterm exam: 20%
- Final exam: 20%

Grading for Graduates

- Five (5) programming assignments: 50%
- Five (5) written homework assignments: 10%
- Midterm exam: **15%**
- Final exam: **15%**
- **One end-of-term survey paper: 10%**

Programming Assignments

- Take a look at the internals of a real OS kernel
- Modify Linux kernel internals
  - Fun!
  - Challenging!
  - Start early!

Late Policy

- Late assignments and homework *are not* accepted
  - Waived for good reasons (e.g., illness, family emergency)
    - Instructor may request documentation (e.g., doctor's note)
- Other exceptions by consent of the instructor
  - E.g., graduate student attending a conference

Regrade Requests

- Must be submitted in writing
- Due within one week of receiving returned, graded assignment

Academic Honesty

- All homework should be done individually
- All programming assignments should be done individually (unless the assignment states otherwise)
- No sharing of code
- Discussion of approaches/kernel APIs okay

Course Website

http://www.cs.rochester.edu/courses/256/fall2014

- Basic course information!
- Lecture slides!
- … and fun for the whole family!

Blackboard

- Programming Assignments!
- Homeworks!
- … and more fun, but just for enrolled students!

The Basics

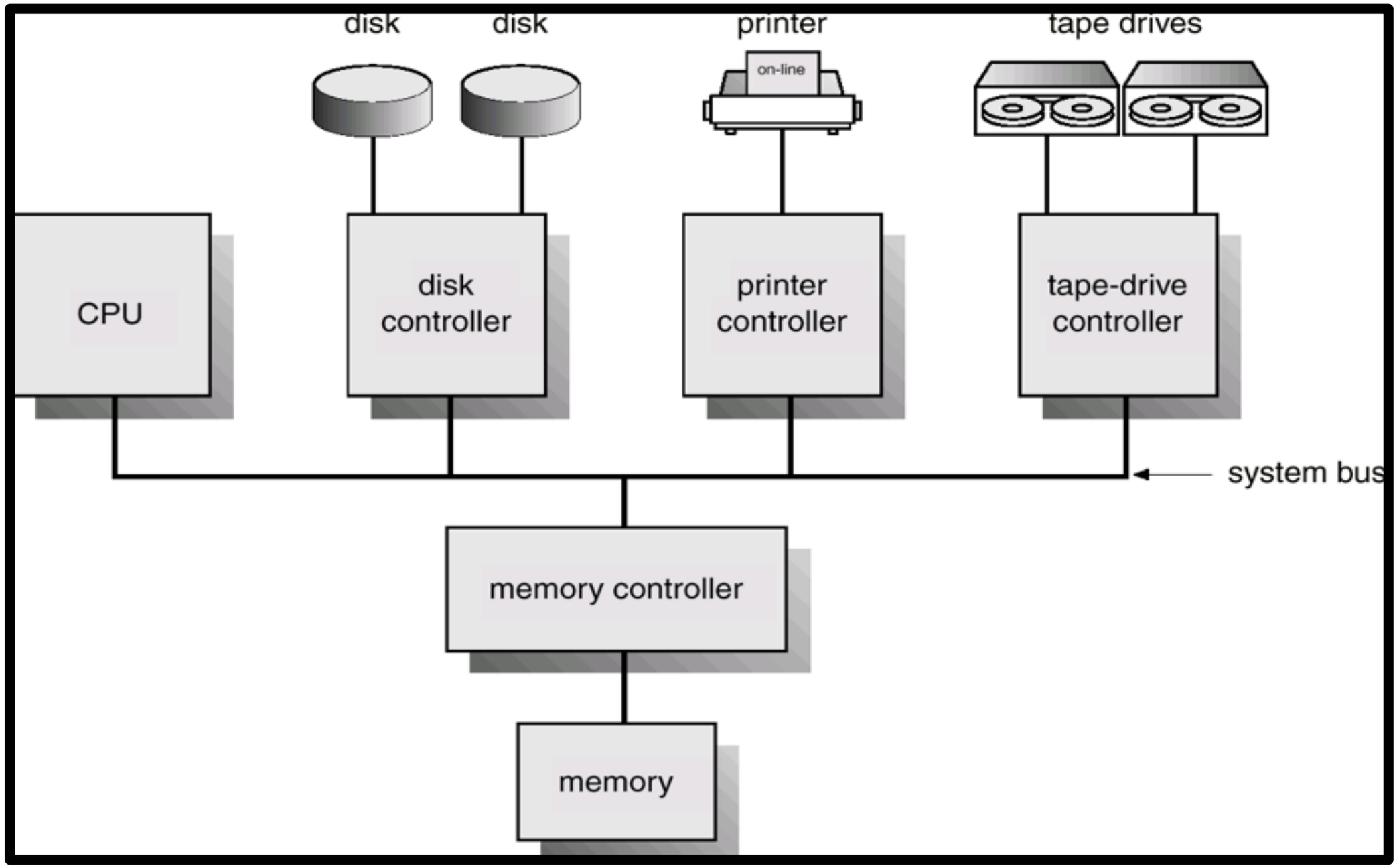
Why Study Operating Systems?

- Concepts applicable to other types of software
  - ***Multi-threaded applications***
  - Web browsers
  - Network servers

Why Study Operating Systems?

- Understand the inner workings of an OS kernel
  - ***Learn to use OS features correctly and efficiently***
  - Learn to develop code that runs within OS kernel
- Learn to design and develop an OS

Computer-System Architecture



What Is an Operating System?

- Software that abstracts the computer hardware
  - Hides messy details of underlying hardware
  - Present resource abstractions that are easy to use
    - Extends or virtualizes underlying machines
- Manages resources
  - Coordinates sharing between different users and programs
  - Examples: processors, disks, networks, timers, mice

Perspectives of the Computer

cut  print

save send

Application Software

System Software

Hardware

malloc()  open()

fork()

Application Software

System Software

Hardware

read-disk  start-printer

push-bits-into-NIC

Application Software

System Software

Hardware

Abstractions

- Processes
  - fork, execve, waitpid, exit, kill
- Files
  - open, close, read, write, seek
- Directories
  - creat, unlink, link, mkdir, rmdir, mount, umount
- Inter-process communication (IPC)
  - pipe, socket, System V IPC (msg, shm, sem)

Reasons for Abstractions

- Reduce functional complexity
- Provide single abstraction over multiple devices
- Resource sharing

Objectives when Sharing Resources

- Efficiency
- Fairness
- Security / protection

How Did We Get Here?

History: Batch Systems

- Computers were large and expensive
- Shared within an organization
- Operator loaded programs for execution
- Multi-tasking
  - When waiting for I/O, start another job!

History: Time Sharing Systems

- Batch systems had
  - Multi-tasking
  - Shared resources (e.g., disk)
- Hook up a bunch of monitors and keyboards
- Multi-tasking == people use the computer simultaneously!

History: Workstations and Microcomputers

- Smaller cheaper systems for just one person
- Multi-tasking provided convenience
- Windowing system
  - Xerox PARC
  - Mac OS 1.0
- Protection provided more stability
- Mac OS X, Windows NT, Linux

Examples of Modern Operating Systems

- Unix family (evolving since 1969)
  - Mac OS X, FreeBSD, Solaris, Linux
- Windows family (evolving since 1989)
  - Windows NT, 2000, 7, and 8
- Mobile operating systems
  - Most based on the Unix family

History: Mobile Systems

- Smaller versions of desktops, but
  - Finite power (i.e., battery)
  - Limited memory
  - Higher reliability demands

Are operating systems done yet?

No!

- Adapting to changes in hardware (e.g., multicore)
- Adapting to new uses
  - Cloud computing
- Reliability (most commercial OSes poorly designed)
- Security
  - Can OS help protect applications?
  - Can we protect the OS from attackers?

Other System Software

- Microkernels
- Exokernels
- Virtual Machines
- Extensible operating systems
- Real time and embedded operating systems
- Safe language operating systems
- And still evolving…

On to the first assignment…

Programming Assignment #1

- Available via Blackboard!
- Completely outside the OS kernel
- Build a shell (command interpreter)
  - Support foreground and background processes
  - Support pipes
  - Support controlling terminal
- Due Sept. 18!

Credits

- Parts of the lecture slides contain original work from Gary Nutt, Andrew S. Tanenbaum, Dave O'Hallaron, Randal Bryant, Kai Shen, and Sandhya Dwarkadas. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).