CIS-4020 Operating Systems

Peter C. Chapin
Vermont Technical College

Goals

- **Learn operating system internals**
- Gain experience with large programs
  - Navigate a large code base written by others
  - Learn how to decipher someone else's code
- We will *not* be covering…
  - Installing, configuring, or managing a system
  - Using operating systems
  - Application programming

Theory vs Practice

- 25% Theory
  - Discuss general topics
- 75% Practice
  - In lab we will write system software
    - Linux kernel programming
    - Modify and extend Phoenix
    - Programming the QNX microkernel system
  - Lots of programming
    - Mostly plain C
    - Some Assembly Language

Five Major Topics

- Five major resources that systems manage
  - Processor time
  - Memory
  - Disk space
  - Devices
  - Network bandwidth
- In this course we will discuss the first four
  - Traditionally don't talk about networking

Labs

- Virtual machine technology
  - Can hack at the OS without disrupting the host
  - Easier to transport experimental system
- Labs are somewhat long and involved
  - Most are multi-week; open ended
  - All will require written reports (except the first)
  - All will be fun and interesting!

LaTeX

- The LaTeX typesetting system is *required* for lab reports
  - Will spend some time covering how to use LaTeX
  - It's a system worth knowing
    - Very high quality output
    - Plays well with software development methodologies
    - Used extensively in academic settings
    - A target output format of other systems (e. g. Doxygen)

Linux

- Complete source code available for study
  - Legally allowed to freely modify it!
- The Linux distribution is over 6 MLOC
  - Finding one's way around is the first challenge
  - We will use the `cscope` code browsing tool
- *We will study 64 bit kernel version 3.18.14*

Phoenix

- Created by VTC Students
  - Nick Guertin, Curtis Aube, Wei Yao Lin
  - Senior project AY 2007/2008
  - Used in this course with permission
  - https://github.com/pchapin/phoenix
- Advantages
  - Microkernel design
  - Small and simple
  - Targets 16 bit “real mode” systems (8086)

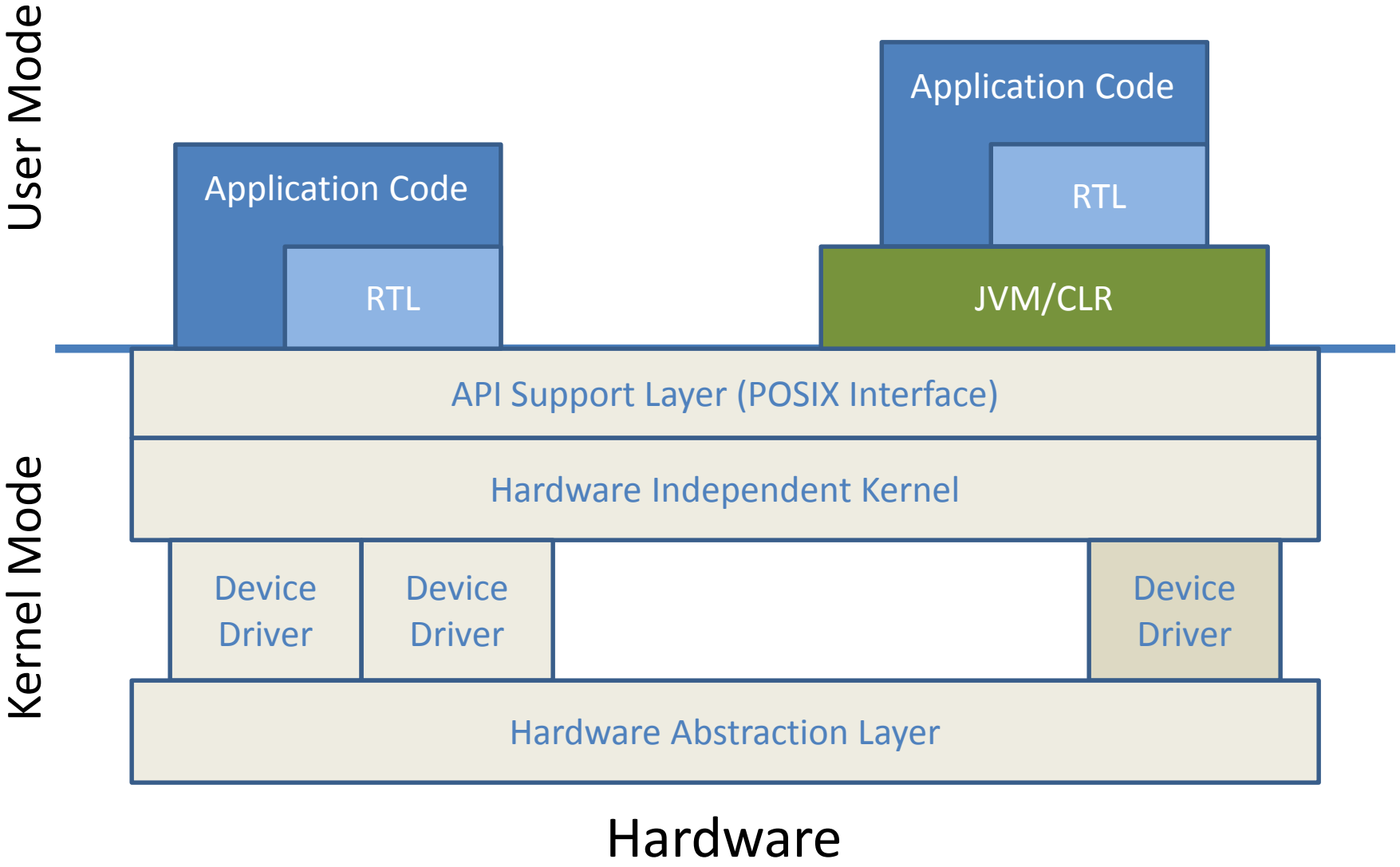
QNX

- Commercial OS for Embedded Systems
  - Targets a variety of platforms
    - Mostly the automotive market
    - Various processors
    - Can be run on the desktop
  - Microkernel design
  - Flexible and well documented
  - Available for free for academic use
  - http://www.qnx.com

Monolithic Kernels

- Linux is monolithic
  - Entire kernel is one large program
  - Different parts of the kernel can call each other directly
  - **Efficient and low overhead**
- *Traditional OS design is for monolithic kernels*

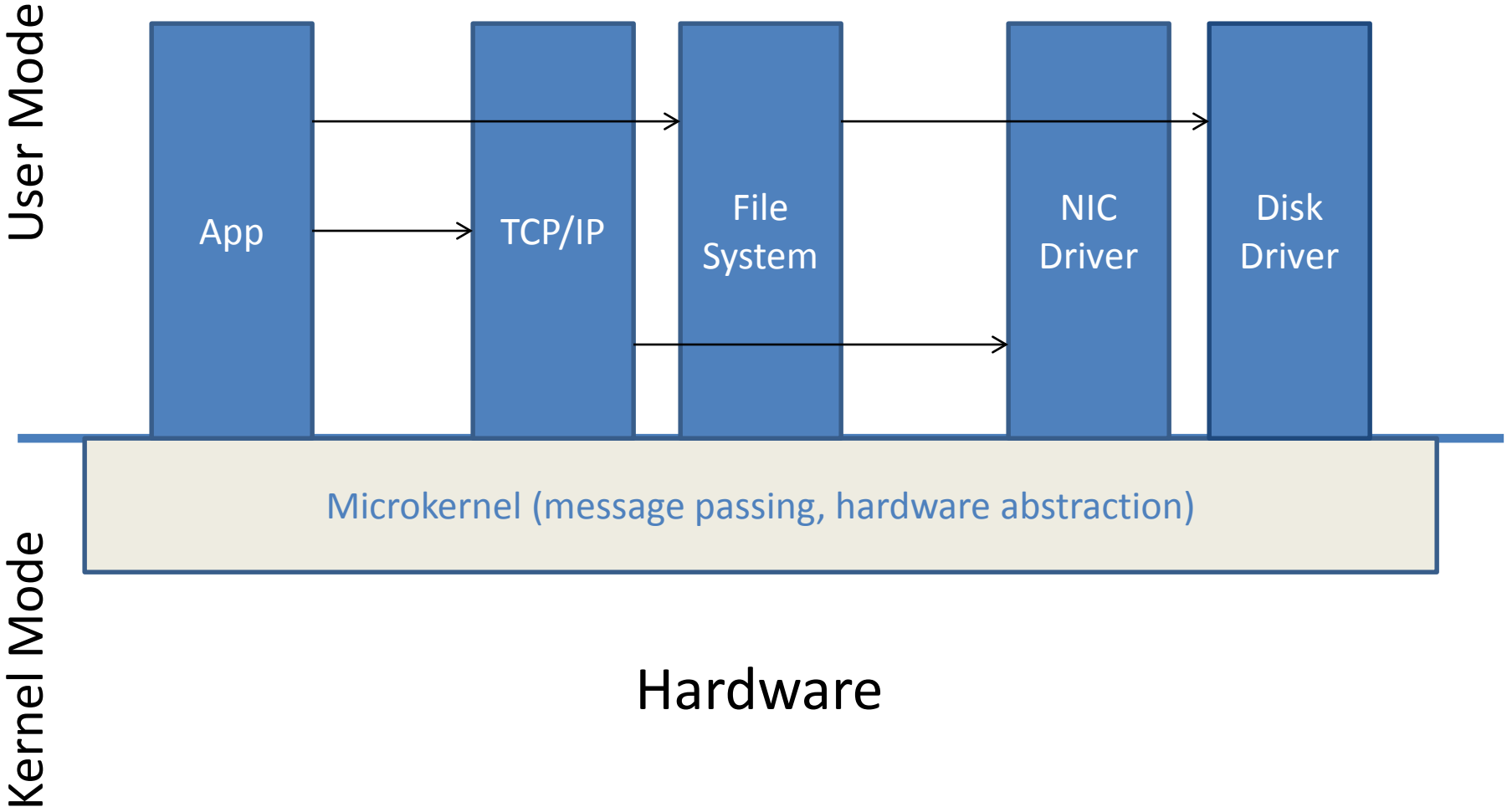
Basic OS (Monolithic) Architecture



Microkernels

- Components of OS in separate processes
  - Components communicate via message passing
  - **More overhead**. Slower
  - **More robust**. If one component crashes the rest of the system continues to run
  - **More flexible**
    - Components can be mixed and matched
    - Components can be (trivially) distributed over a network
    - Components can be written in any programming language
- *Most OS research uses microkernels*

Basic OS (Microkernel) Architecture



Windows?

- We won't discuss Windows in this course
  - No source code available
  - No new architectural issues of significance
    - Windows is a microkernel internally but wraps all OS components into a monolithic lump for performance reasons
- Nothing wrong with Windows
  - It's a modern system with a reasonable design
  - Not different enough from Linux to warrant special study

Android?

- We won't discuss Android in this course
  - Modified Linux
  - Essentially all differences in the UI and libraries
  - From a kernel perspective…
    - … not different enough from stock Linux to warrant separate handling.

Real Time Operating Systems

- **If a computation does not complete in the specified time, then the system has failed**.
- Many embedded systems have real time requirements
- Special OS support is needed
  - Arbitrary multi-tasking makes it impossible for a program to always meet deadlines
- Special programming methods needed
  - Programmer must take into account time issues

Distributed Operating Systems

- Spread the OS over multiple machines
  - Many open questions; systems not widely used
  - Interesting and may be important in the future
  - Clustering is a simple form of distributed OS
- Distributed OS…
  - Applications automatically distributed by the OS. *Applications unaware of their distributed nature*.
- Distributed applications…
  - Applications manage their distribution. *OS unaware of the application's nature.*

Operating Systems Landscape

- Desktop/Server systems
  - Fairly stark. Unix and Windows dominate
  - Servers a little more interesting: clustering, scalability, advanced file systems
- Embedded systems
  - This is where the action is
  - Many, *many* different systems
  - Some very small (**Salvo** can run with zero RAM!)
  - Some are full scale systems adapted for embedded work

Class Organization

- Traditional Delivery
  - Face to face lectures and labs
- Course materials on my web site
  - http://web.vtc.edu/users/pcc09070/cis-4020
  - Grading policy, late policy, etc
  - Handouts, assignments, useful links
  - Your first assignment is already posted
- Homework/Lab submissions on Moodle

Good Luck!

And don’t forget to have fun!