Microkernels

Peter Chapin
Vermont State University

Linux is Monolithic

- A single large “executable”
  - Every function can call every other function (in principle)
    - Except... there are complications with modules.
  - Global data is global to the entire kernel.
- Pros
  - High efficiency. Different kernel components have direct access to each other.
- Cons
  - Less flexible, less robust.
  - Development is complicated.

Microkernels are Different

- Kernel only supports:
  - Thread scheduling.
  - A fast method of IPC (Interprocess Communication).
  - Interrupt entry points.
- All other functions are in ordinary processes:
  - Device drivers
  - File systems
  - Network protocols
  - Memory management algorithms

Microkernel Pros

- Flexible
  - All components modularized for easy mix-n-match.
  - Distribution over a network “for free.”
  - Development is easier.
    - Components can be written in any language.
    - Built and debugged with ordinary development tools.
- Robust
  - Failure of one component is isolated.
  - Microkernel is easier to verify.
  - Constrained systems are straightforward to configure.

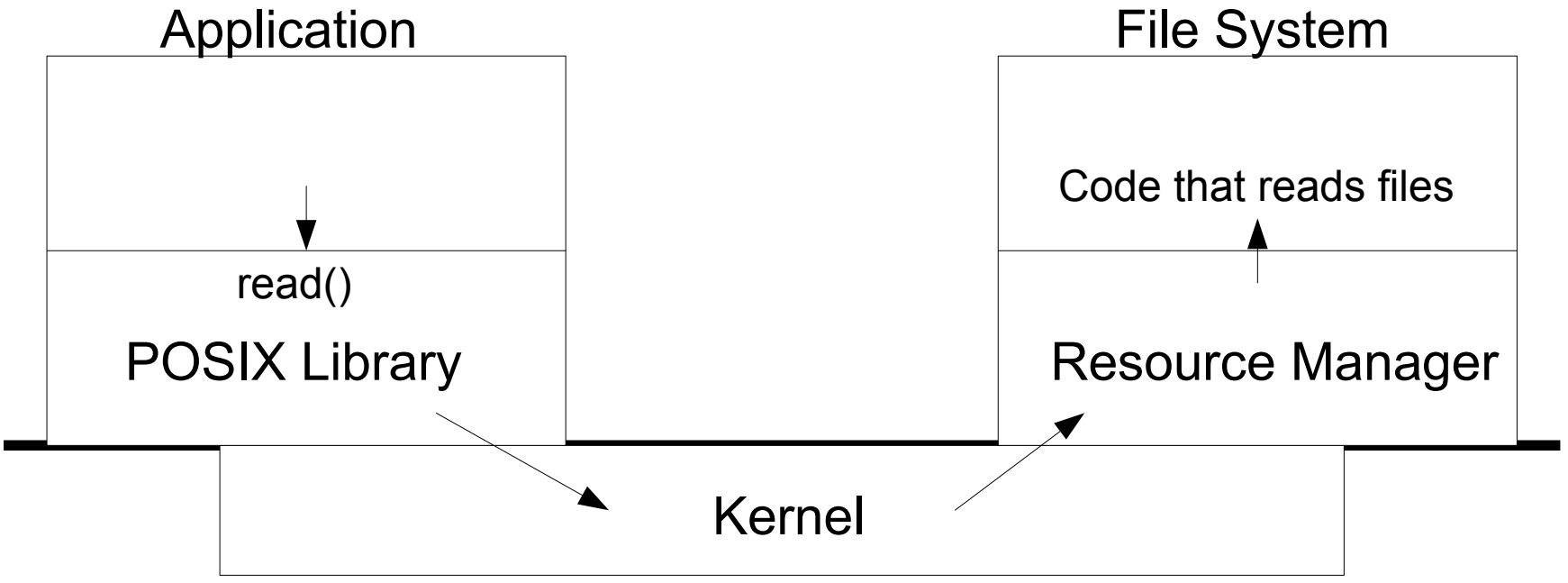
Microkernel Cons

- Slow
  - When components communicate, many user/kernel transitions occur as messages are sent back and forth through the microkernel.
  - Lots of overhead!
- Show stopper for many users.
  - Most experimental systems are microkernel based.
  - Most commercial systems are monolithic.

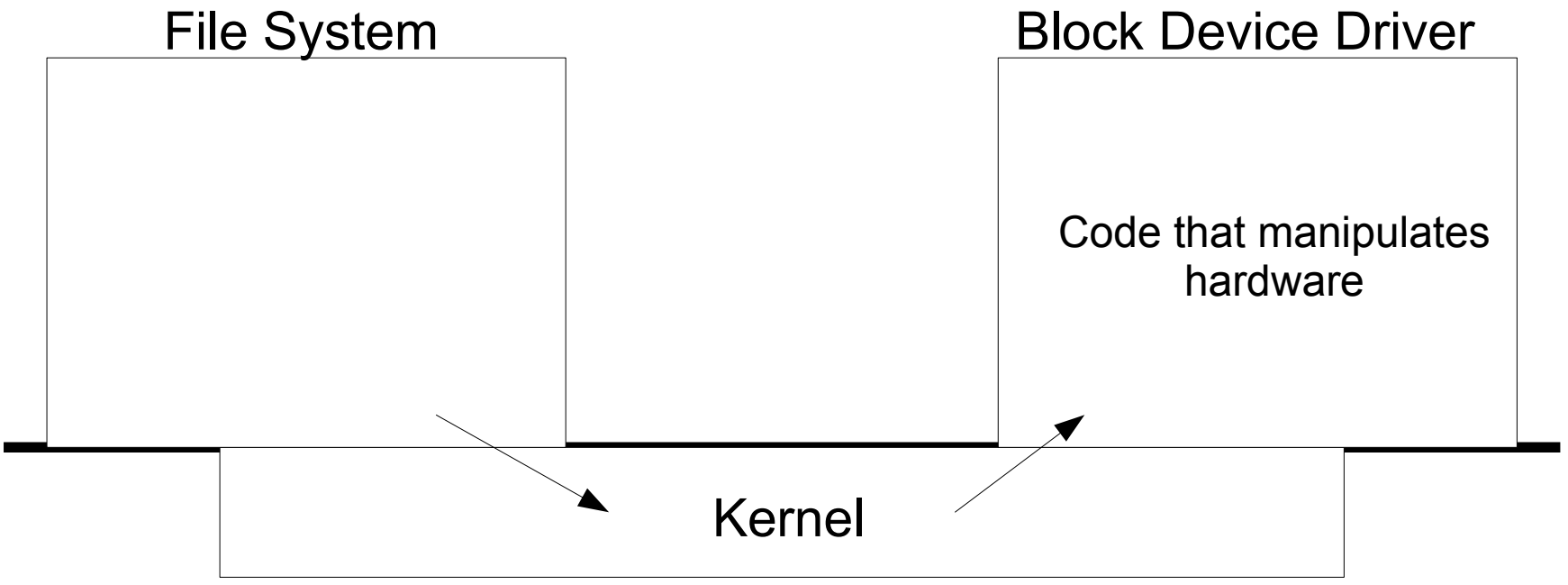
QNX

- QNX is a commercial microkernel system.
  - Targets embedded market.
    - High modularity is important.
    - Robustness is important.
  - Could, in the past, be used as a desktop system.
  - Source code for kernel recently published.
- Implements the POSIX API
  - Non-trivial: kernel only understands messages.
- Also offers real time support (separate slides!)

POSIX Support on QNX



Low Level Access on QNX



Serial Driver Lab

- You will write a driver for the serial port.
  - Interact with Resource Manager interface.
    - Allows your driver to be accessed via POSIX functions open(), read(), etc, from other processes.
  - Interact with the 16550 UART.
    - Configure hardware.
    - Handle interrupts.
- Driver is multi-threaded
  - One thread to manage UART.
  - One thread to wait for messages.

Serial Driver Block Diagram