Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance

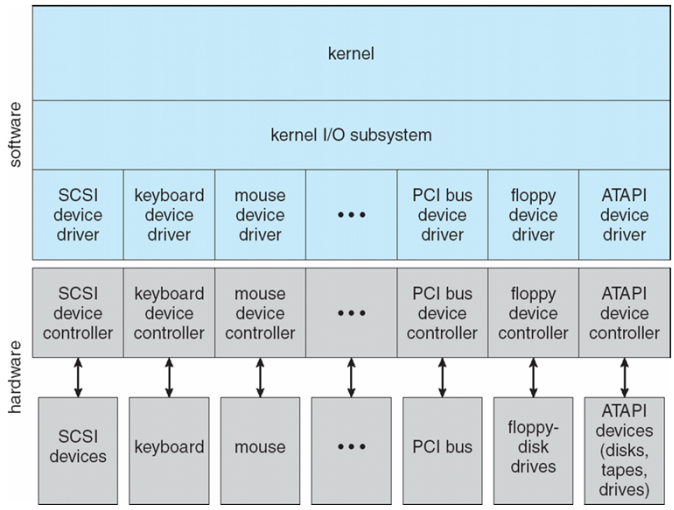
(slide adjusted by R. Doemer, 02/24/11)

Application I/O Interface

- **I/O system calls** encapsulate device behaviors in generic classes
- **Device-driver layer** hides differences among I/O controllers from kernel
- **Devices** vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

(slide modified by R. Doemer, 03/01/11)

Kernel I/O Structure



(slide modified by R. Doemer, 03/01/11)

Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Block and Character Devices

- **Block devices** include disk drives
  - Commands include **read**, **write**, **seek**
  - **Raw I/O** or **file-system** access
  - **Memory-mapped file** access possible
- **Character devices** include keyboards, mice, serial ports
  - Commands include **`get, put`**
    - e.g. Nachos: raw Console
  - Libraries layered on top allow line editing
    - e.g. Nachos: Synchronous Console

(slide modified by R. Doemer, 03/01/11)

Network Devices

- **Network Devices**
  - Vary enough from block and character to have **own interface**
- Unix and Windows NT/9x/2000 include **socket interface**
  - Separates network protocol from network operation
  - Includes **`select`** functionality
    - Synchronous I/O multiplexing
- Approaches vary widely
  - **Pipes**
  - FIFOs
  - Streams
  - **Queues**
  - **Mailboxes**

(slide modified by R. Doemer, 03/01/11)

Clocks and Timers

- **Clocks** and **Timers**
  - Provide current time, elapsed time, timer
- **Programmable interval timer** used for timings, periodic interrupts
- **I/O Control** function covers odd aspects of I/O such as clocks and timers
  - `ioctl` on UNIX: device-specific control functions

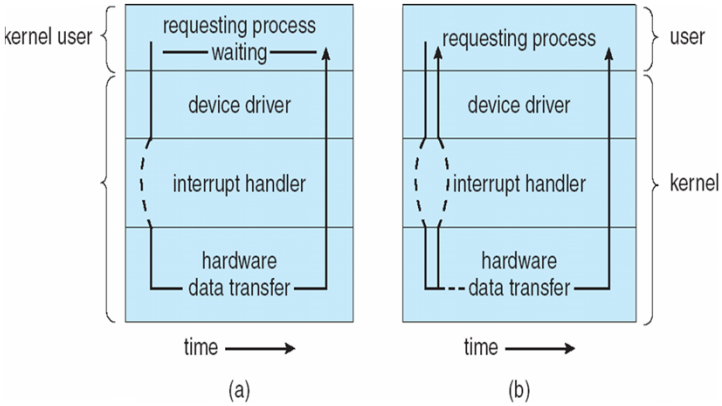
(slide modified by R. Doemer, 03/01/11)

Blocking and Nonblocking I/O

- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed
  - e.g. Nachos: see **Console** vs. **SynchConsole**

(slide modified by R. Doemer, 03/01/11)

Two I/O Methods



Synchronous I/O

e.g. Nachos **`SynchConsole`**

Asynchronous I/O

e.g. Nachos **`Console`**

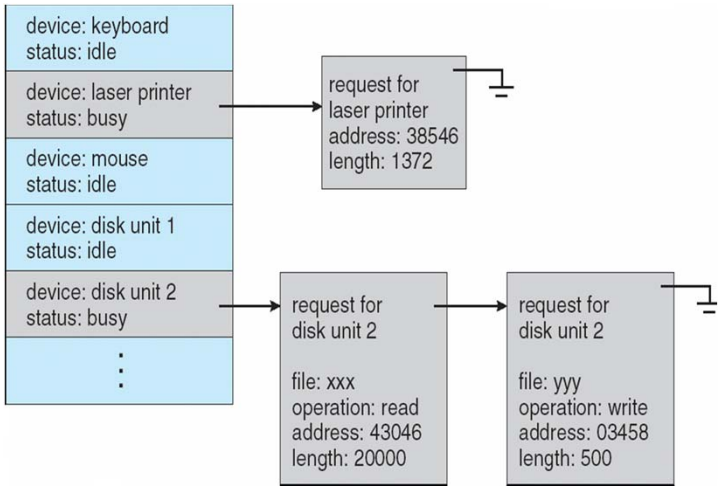
(slide modified by R. Doemer, 03/01/11)

Kernel I/O Subsystem

- **I/O Scheduling**
  - Some I/O request ordering via **per-device queue**
  - Some operating systems try fairness
- **Buffering** - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”

(slide modified by R. Doemer, 03/01/11)

Device-Status Table

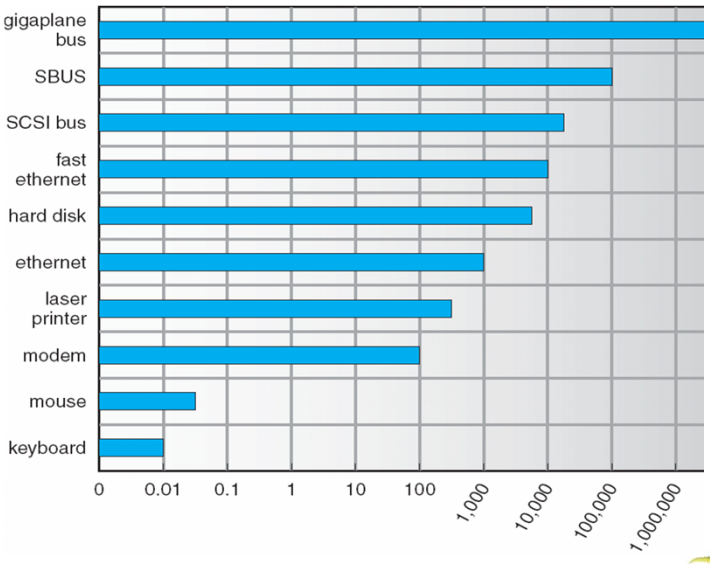


Kernel I/O Subsystem

- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- **Spooling** - hold output for a device
  - If device can serve only one request at a time
  - i.e. printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

(slide modified by R. Doemer, 03/01/11)

Sun Enterprise 6000 Device-Transfer Rates



(slide left un-modified by R. Doemer, 03/01/11)

Error Handling

- OS can recover from certain **I/O errors**
  - disk read
  - device unavailable
  - transient write failures
- Unrecoverable errors return an **error number** or code when I/O request fails
- **System error logs** hold problem reports

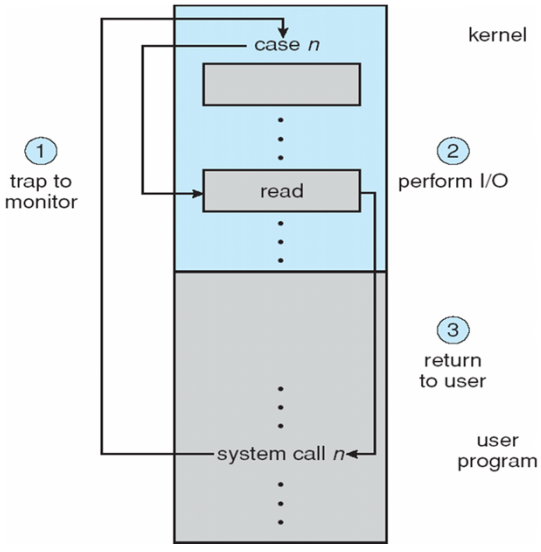
(slide modified by R. Doemer, 03/01/11)

I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - **All I/O instructions defined to be privileged**
  - I/O must be performed via **system calls**
    - Memory-mapped and I/O port memory locations must be protected too

(slide modified by R. Doemer, 03/01/11)

Use of a System Call to Perform I/O

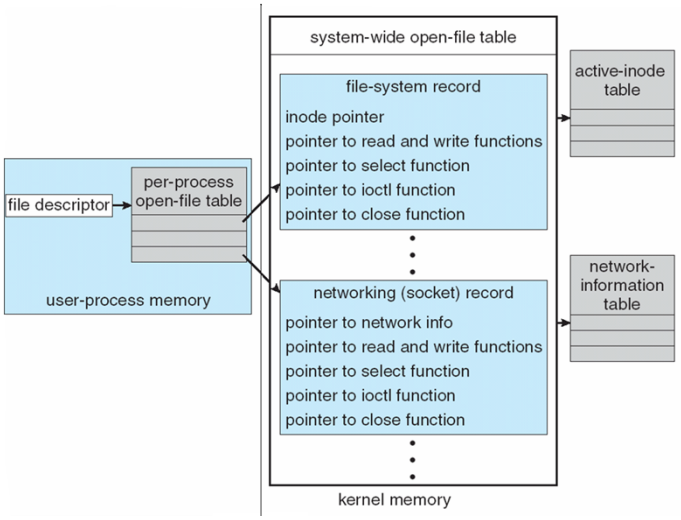


Kernel Data Structures

- Kernel keeps **state info** for I/O components, including
  - open file tables,
  - network connections,
  - character device state
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks, …
- Some use object-oriented methods and message passing to implement I/O

(slide modified by R. Doemer, 03/01/11)

UNIX I/O Kernel Structure

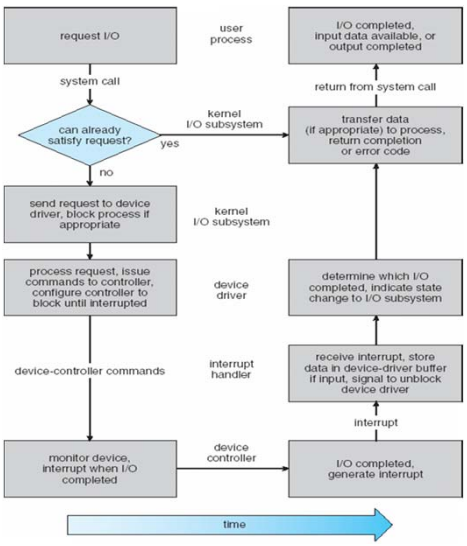


I/O Requests to Hardware Operations

- Consider steps in **reading a file from disk** for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

(slide modified by R. Doemer, 03/01/11)

Life Cycle of An I/O Request

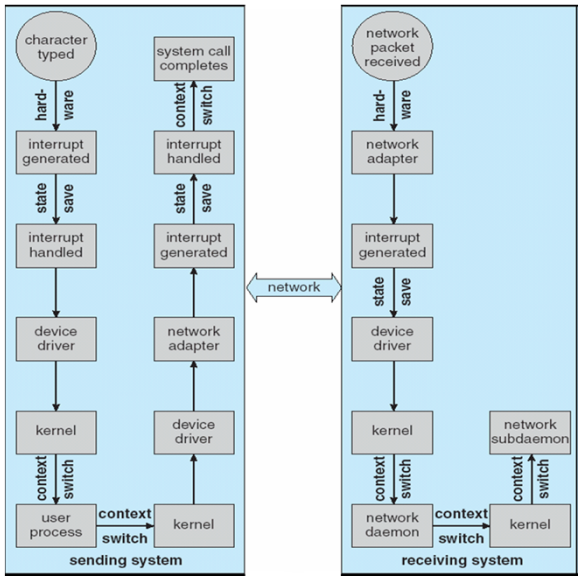


Performance

- I/O is a **major factor** in **system performance**:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic is especially stressful

(slide modified by R. Doemer, 03/01/11)

Intercomputer Communications



Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using
  - large transfers,
  - smart controllers,
  - polling
  - DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

(slide modified by R. Doemer, 03/01/11)

End of Chapter 13