CSE 4/521 Introduction to Operating Systems

Lecture 24 – I/O Systems

(Overview, Application I/O Interface, Kernel I/O Subsystem)

Summer 2018

Overview

Objective:

- Explore the structure of an operating system's I/O subsystem.

- Overview
- Application I/O Interface
- Kernel I/O Subsystem (half)

Recap

- Allocation Methods
  1. Contiguous Allocation
  2. Linked Allocation
  3. Indexed Allocation
- Free-Space Management
  1. Bit vector Free-Space Management
  2. Linked Free-Space List

Questions

1. Contrast the performance of the 3 techniques for allocating disk blocks (contiguous, linked, and indexed) for both sequential and random file access. (Easy)

2. What are the advantages of the variant of linked allocation that uses a FAT to chain together the blocks of a file? (Medium)

3. Why must the bit map for free-space allocation be kept on mass storage, rather than in main memory? (Easy)

Overview

- Overview
- Application I/O Interface
- Kernel I/O Subsystem (half)

Overview

- I/O management is a major component of operating system design and operation
  - I/O devices vary greatly
  - Various methods to control them
  - Performance management
  - New types of devices frequent
- Ports, busses, device controllers connect to various devices
- Device drivers encapsulate device details
  - Present uniform device-access interface to I/O subsystem

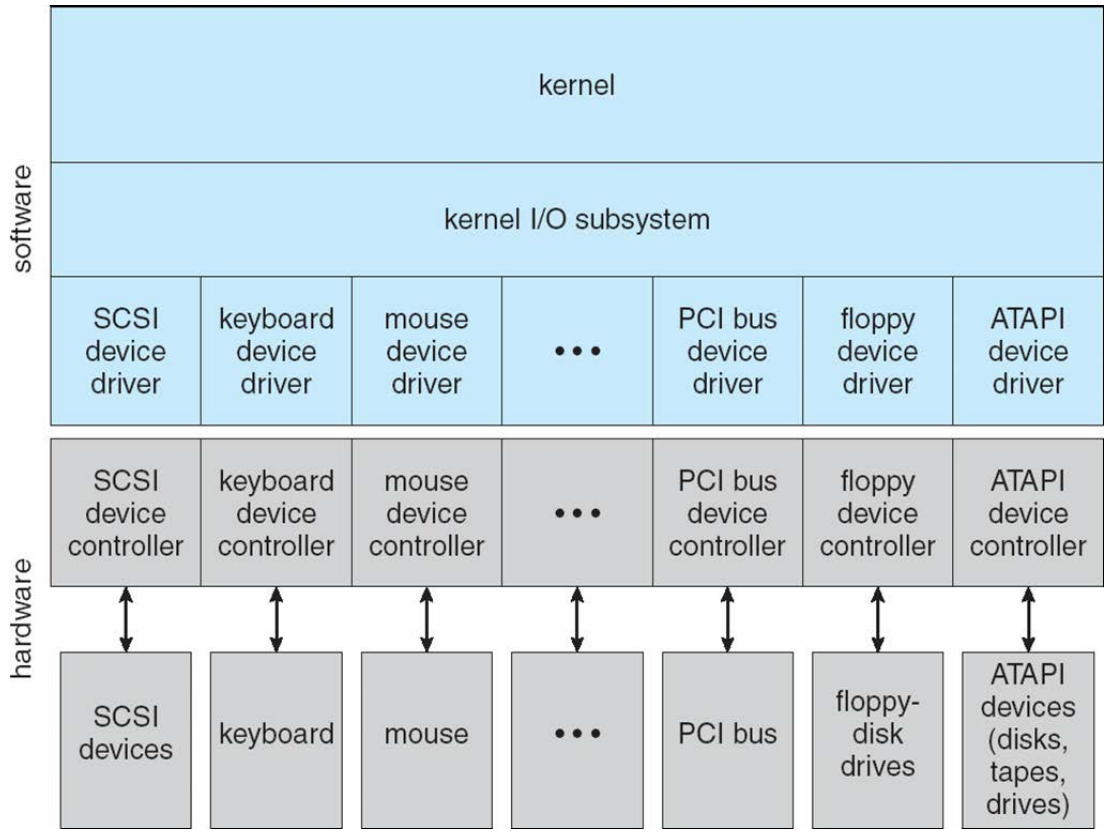
Overview

- Overview
- Application I/O Interface
- Kernel I/O Subsystem (half)

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
- New devices talking already-implemented protocols need no extra work
- Each OS has its own I/O subsystem structures and device driver frameworks
- Devices vary in many dimensions
  - Character-stream or block
  - Sequential or random-access
  - Synchronous or asynchronous (or both)
  - Sharable or dedicated
  - Speed of operation
  - read-write, read only, or write only

Application I/O Interface: A Kernel I/O Structure



Application I/O Interface: 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

Application I/O Interface: Characteristics of I/O Devices

- Subtleties of devices handled by device drivers
- Broadly I/O devices can be grouped by the OS into
  - Block I/O
  - Character I/O (Stream)
  - Memory-mapped file access
  - Network sockets
- For direct manipulation of I/O device to access specific characteristics :
  - Unix ioctl() call to send arbitrary bits to a device control register and data to device data register

Application of I/O Interface: Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O, direct I/O, or file-system access
  - Memory-mapped file access possible
    - File mapped to virtual memory and clusters brought via demand paging
  - DMA
- Character devices include keyboards, mice, serial ports
  - Commands include get(), put()
  - Libraries layered on top allow line editing

Application of I/O Interface: Network Devices

- Varying enough from block and character to have own interface
- Linux, Unix, Windows and many others include socket interface
  - Separates network protocol from network operation
  - Includes select() functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

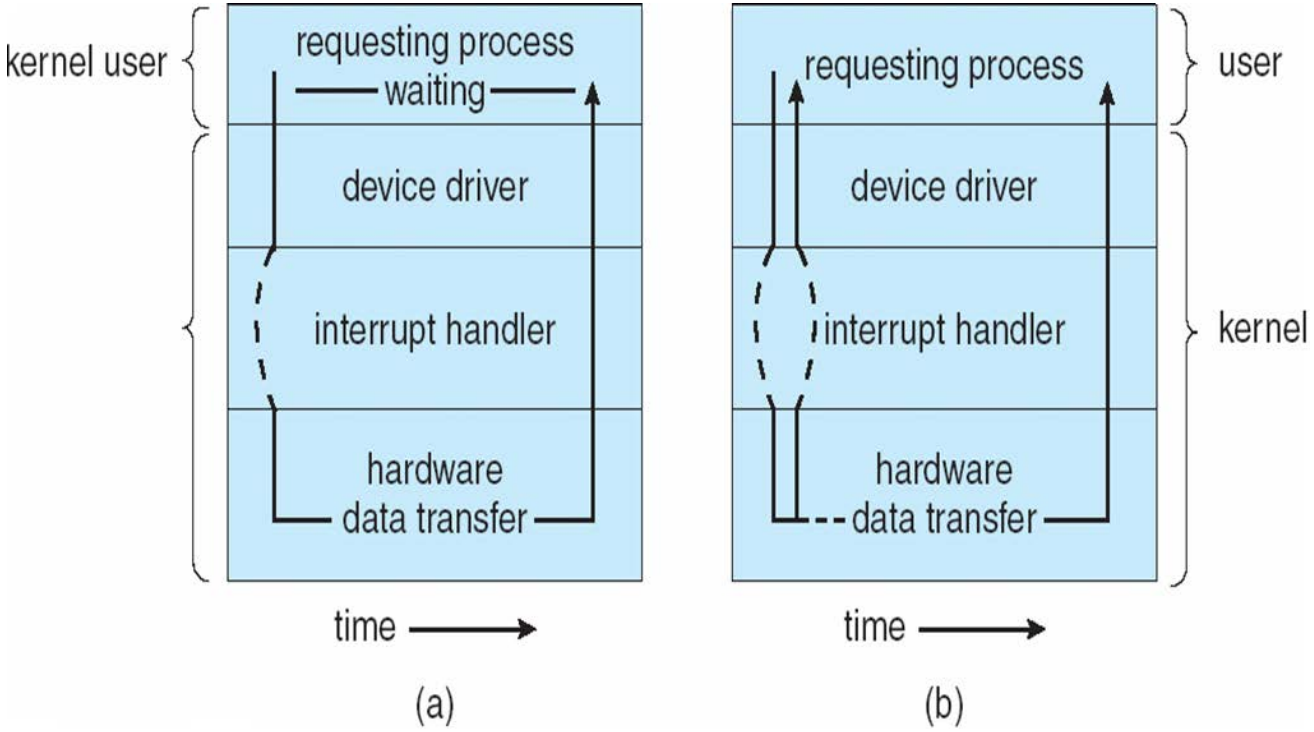
Application of I/O Interface: Clocks and Timers

- Provide current time, elapsed time, timer
- Normal resolution about 1/60 second
- Some systems provide higher-resolution timers
- Programmable interval timer used for timings, periodic interrupts
- ioctl() (on UNIX) covers odd aspects of I/O such as clocks and timers

Application of I/O Interface: Nonblocking and Asynchronous 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
  - `select()` to find if data ready then `read()` or `write()` to transfer
- Asynchronous - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

Application of I/O Interface: Two I/O Methods



Synchronous

Asynchronous

Application of I/O Interface: Vectored I/O

- Vectored I/O allows one system call to perform multiple I/O operations
- For example, Unix `readve()` accepts a vector of multiple buffers to read into or write from
- This scatter-gather method better than multiple individual I/O calls
  - Decreases context switching and system call overhead
  - Some versions provide atomicity
    - Avoid for example worry about multiple threads changing data as reads / writes occurring

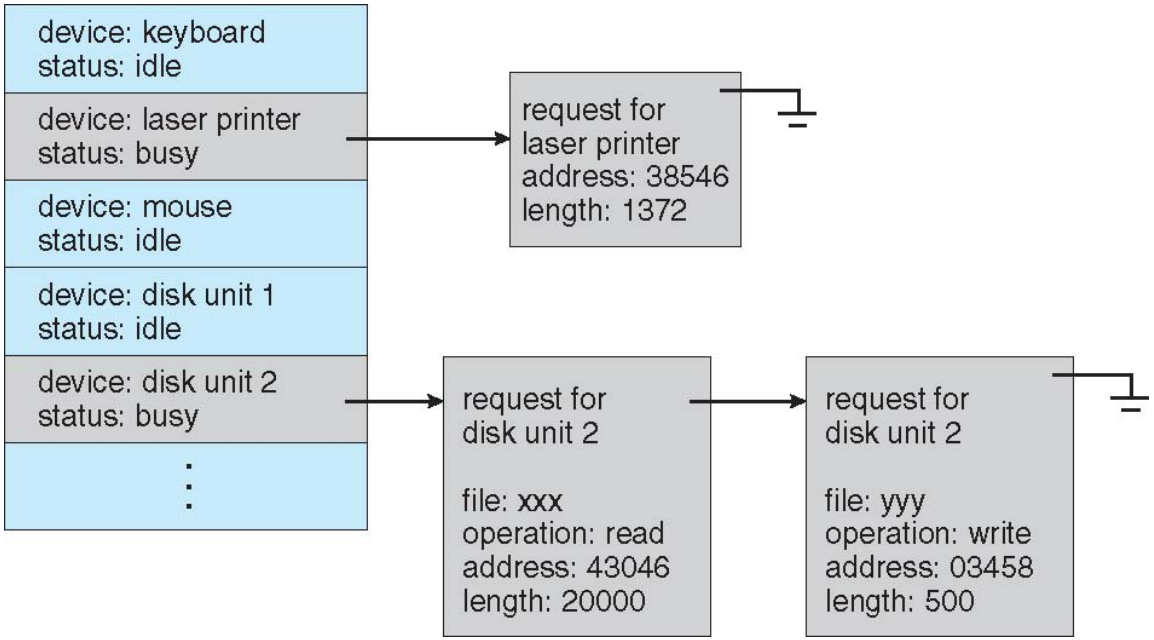
Overview

- Overview
- Application I/O Interface
- Kernel I/O Subsystem (half)

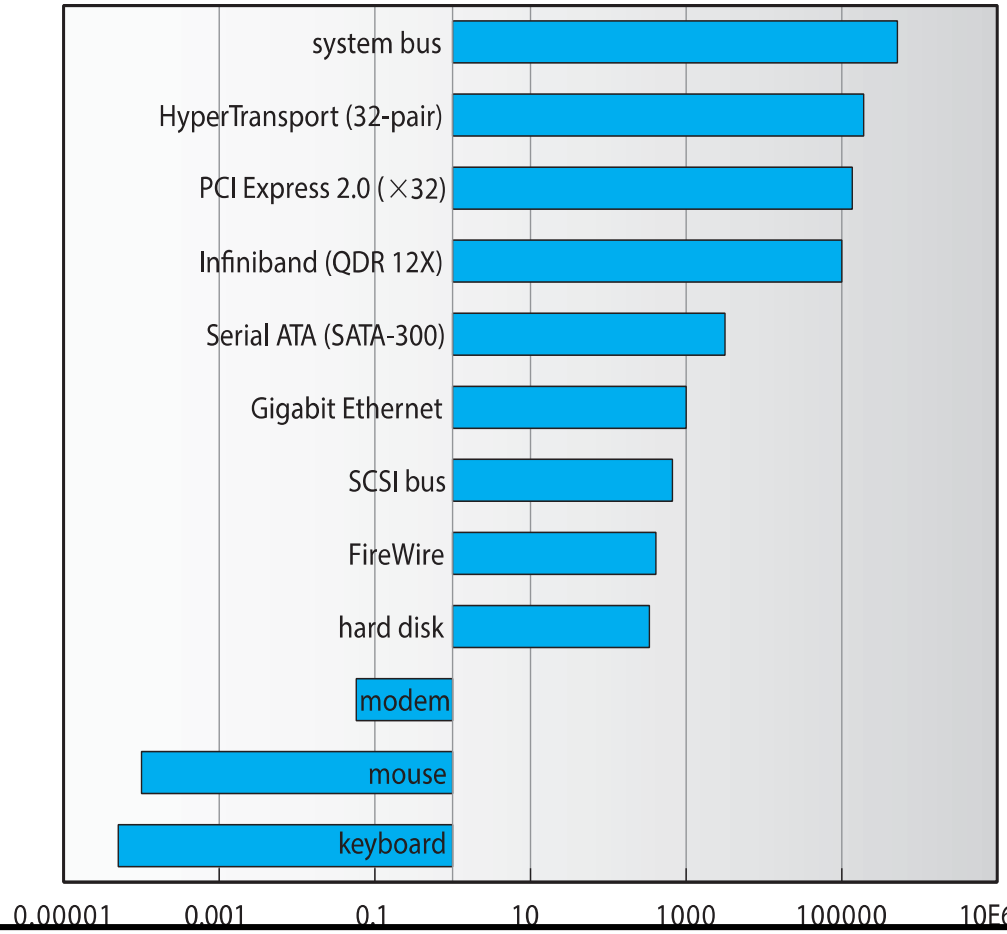
Kernel I/O Subsystem

- Scheduling
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
  - Some implement Quality Of Service (i.e. IPQOS)
- 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”
  - Double buffering – two copies of the data
    - Kernel and user
    - Varying sizes
    - Full / being processed and not-full / being used
    - Copy-on-write can be used for efficiency in some cases

Kernel I/O Subsystem: Device-status Table



Kernel I/O Subsystem: Sun Enterprise 6000 Device-Transfer Rates



Kernel I/O Subsystem

- Caching - faster device holding copy of data
  - Always just a copy
  - Key to performance
  - Sometimes combined with buffering
- 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 de-allocation
  - Watch out for deadlock

Kernel I/O Subsystem: Error Handling

- OS can recover from disk read, device unavailable, transient write failures
  - Retry a read or write, for example
  - Some systems more advanced – Solaris FMA, AIX
    - Track error frequencies, stop using device with increasing frequency of retry-able errors
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

Credits for slides

Silberschatz, Galvin and Gagne