Chapter 13: I/O Systems

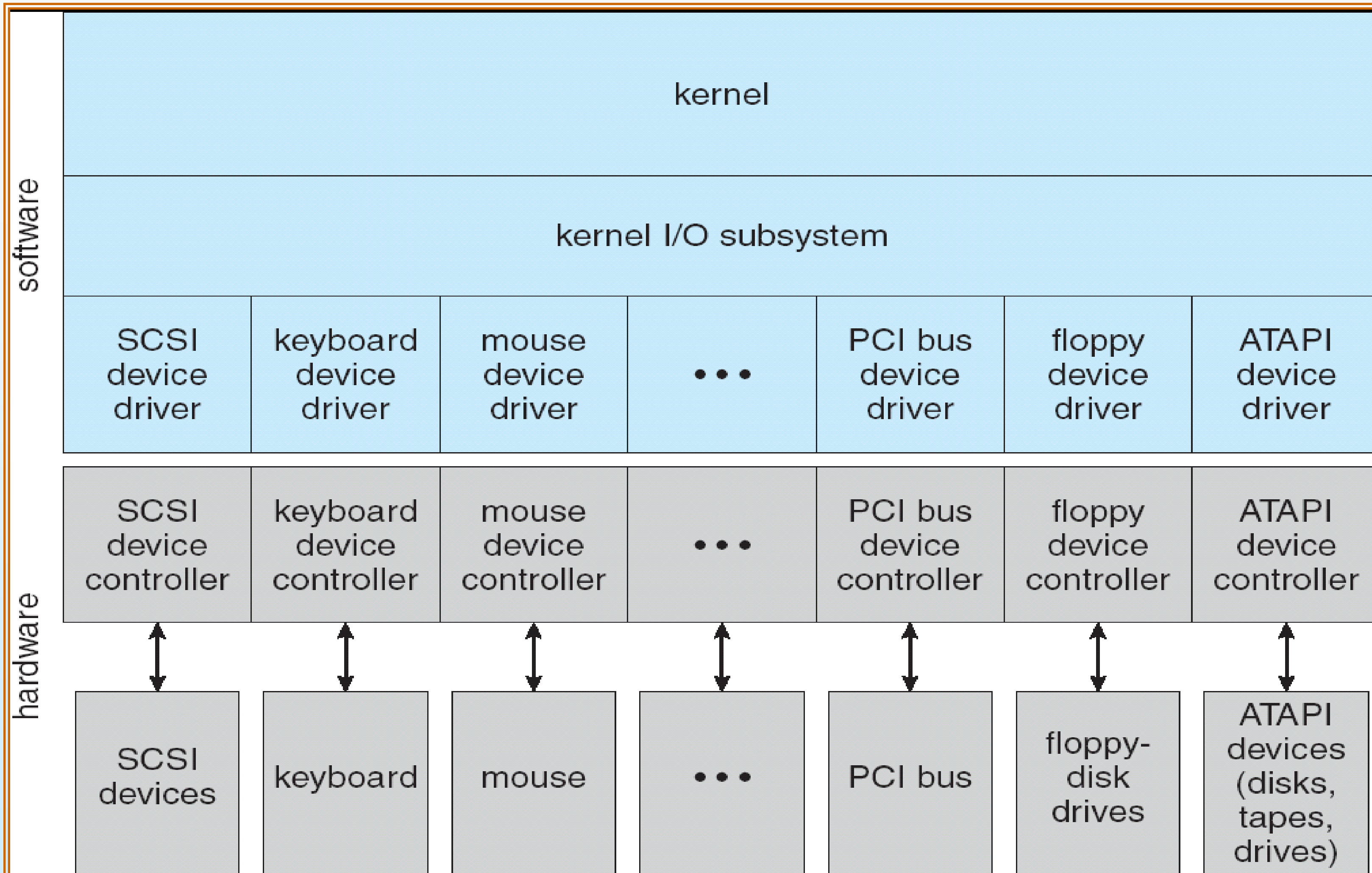
Chapter 13: I/O Systems

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

13.1 Overview

- I/O management is a major component of operating system design and operation
  - Important aspect of computer 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

A Kernel I/O Structure



Overview

- Varied methods needed to control I/O devices，because they vary so widely in their function and speed.
- These methods form the I/O subsystem of the kernel, which **separates the rest of the kernel from** the **complexities of I/O devices**.
- I/O-device technology exhibits two conflicting trends
  - Standardization of software and hardware interfaces；
  - Broad variety of I/O devices；

Overview—device driver

- To encapsulate the details and oddities of different devices, the kernel of an operating system is structured to use **device-driver modules**.
- **Device drivers**
  - Present a uniform device access interface to the I/O subsystem for all kinds of different I/O devices.
  - 系统为不同的设备设计了不同的设备驱动程序
    - 对于不同硬件设备，为I/O子系统提供了统一的设备访问接口
    - 根据I/O子系统的要求完成对硬件设备的具体访问
    - 是硬件设备和系统之间的桥梁
    - 简化了I/O子系统的设计
  - 类似于VFS中的虚拟文件接口层(VFS Interface)，为不同的文件系统提供了统一的文件系统调用接口
  - such as system calls provide a standard interface between the application and the operating system.

Overview--ioctl

- 应用程序可以使用内核提供的统一接口访问I/O设备；
- **统一访问接口的使用，方便了内核及应用程序的设计与编码，但也导致应用程序无法使用设备的具体特性，降低了设备的性能；**
- **UNIX还提供了一个系统调用Ioctl()，用户可以通过该系统调用直接通过设备驱动程序操纵I/O设备；**
- **`ioctl`** (on UNIX) covers odd aspects of I/O
  - Ioctl() – I/O Control
  - Ioctl() can transparently passes arbitrary commands from an application to a device drive
  - The ioctl() system call enables an application to access any functionality that can be implemented by any device driver, without the need to invent a new system call

13.2 I/O Hardware

- Incredible variety of I/O devices
  - Storage
  - Transmission
  - Human-interface

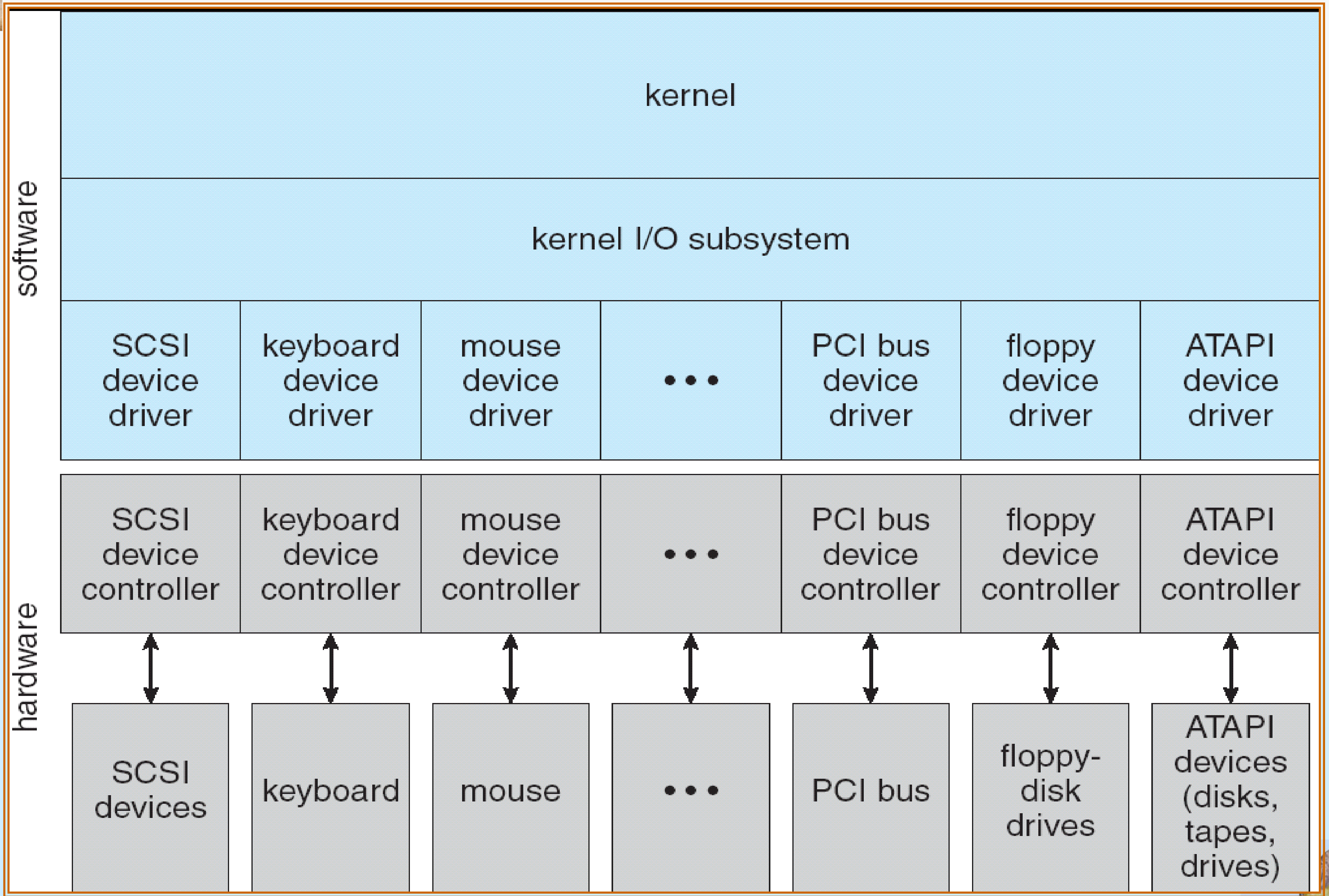
I/O Hardware (Cont.)

- Common concepts – signals from I/O devices interface with computer
  - **Port** – connection point for device (Status, Control, Data(I/O))
  - **Bus** - **daisy chain** or shared direct access
    - **PCI** bus common in PCs and servers, PCI Express (**PCIe**)
    - **expansion bus** connects relatively slow devices
  - **Controller** (**host adapter**) – a collection of electronics that operate port, bus, device
    - Sometimes integrated
    - Sometimes separate circuit board (host adapter)
    - Contains processor, microcode, private memory, bus controller, etc
      - Some talk to per-device controller with bus controller, microcode, memory, etc

12.3 Application I/O Interface

- Like other complex software-engineering problems, the approach here involves **abstraction**, **encapsulation**, and **software layering**
  - See next page -A Kernel I/O Structure
- **Abstract** away the detailed differences in I/O devices by identifying a few general kinds. Each general kind is accessed through a standardized set of functions---an interface.
- The differences are **encapsulated** in kernel modules called **device drivers** that internally are custom-tailored to each device but that export one of the standard interfaces.

A Kernel I/O Structure



Application I/O Interface (Cont.)

- **I/O system calls** encapsulate device behaviors in a few generic classes that **hide hardware differences** from **applications**.
  - **I/O**系统调用为应用程序提供了统一的调用接口，隐含了硬件设备的不同（由**I/O**子系统负责处理对不同设备的访问）
- **Device-driver layer** hides differences among I/O controllers from the I/O subsystem of the kernel
  - 设备驱动程序层为**I/O**子系统提供统一的访问接口，隐含了**I/O**控制器的不同
- Making the I/O subsystem **independent of** the hardware simplifies the job of the operating-system developer.

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

操作系统中的I/O子系统通常由四个层次组成，每一层明确定义了与邻近层次的接口。其合理的层次组织顺序是（）。

A 用户级I/O软件、设备无关性软件、设备驱动程序、中断处理程序

B 用户级I/O软件、设备无关性软件、中断处理程序、设备驱动程序

C 用户级I/O软件、设备驱动程序、设备无关性软件、中断处理程序

D 用户级I/O软件、 中断处理程序、 设备无关性软件、设备驱动程序

提交

续上页

I/O请求

I/O应答

用户层软件	产生I/O请求、格式化I/O、SPOOLing 实现与用户交互的接口，用户直接调用在用户层提供的与I/O操作有关的库函数，操作I/O设备
设备独立性软件	映射、保护、分块、缓冲、分配 实现与设备驱动器的统一接口、设备命名、设备保护以及设备的分配与释放等，为数据传送提供必要的存储空间
设备驱动程序	设置设备寄存器，检查寄存器内容 与硬件直接相关，负责具体实现系统对设备发出的操作指令，驱动I/O设备
中断处理程序	用于保护被中断进程的CPU环境，转入相应的中断处理程序进行处理，处理完成后再恢复到被中断进程的现场后返回到被中断的进程
硬件	

I/O系统的层次及功能

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

用户发出磁盘I/O请求后，系统的处理流程是：用户程序→系统调用处理程序→设备驱动程序→中断处理程序。其中，计算数据所在磁盘的柱面号、磁头号、扇区号的程序是（）。

A 用户程序

B 系统调用处理程序

C 设备驱动程序

D 中断处理程序

提交

Devices vary in many dimensions(page 507)

- **Character-stream or block**
  - A character-stream device transfers bytes one by one,
  - A block device transfers a block of bytes as a unit.
- **Sequential or random-access**
  - A sequential device transfers data in a fixed order determined by the device
  - The user of a random-access device can instruct the device to seek to any of the available data storage locations
- **Synchronous or asynchronous**
  - A synchronous device performs data transfers with predictable response times.
  - An asynchronous device exhibits irregular or unpredictable response times.

Devices vary in many dimensions（page 507）

- **Sharable or dedicated**
  - A sharable device can be used concurrently by several processes or threads; a dedicated device cannot.
- **Speed of operation**
  - Device speeds range from a few bytes per second to a few gigabytes per second.
- **Read-write, read only, or write only**
  - Some devices perform both input and output, but others support only one data direction.

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

13.3.1 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
  - Libraries layered on top allow line editing

13.3.3 Clocks and Timers

- Provide current time, elapsed time, timer
- **Programmable interval timer** used for timings, periodic interrupts
- ioctl (on UNIX) covers odd aspects of I/O such as clocks and timers

13.3.4 Blocking and Nonblocking I/O

- **Blocking - process suspended (blocked) 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) (between two devices)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - A nonblocking read() return immediately **with whatever data are available**– the full number of bytes requested, fewer, or none at all.
- **Asynchronous - process runs while I/O executes**
  - An alternative nonblocking I/O;
  - An asynchronous read0 call requests a transfer that will be performed in its entirety but that **will complete at some future time.**
  - **I/O subsystem signals process when I/O completed;**
  - Difficult to use

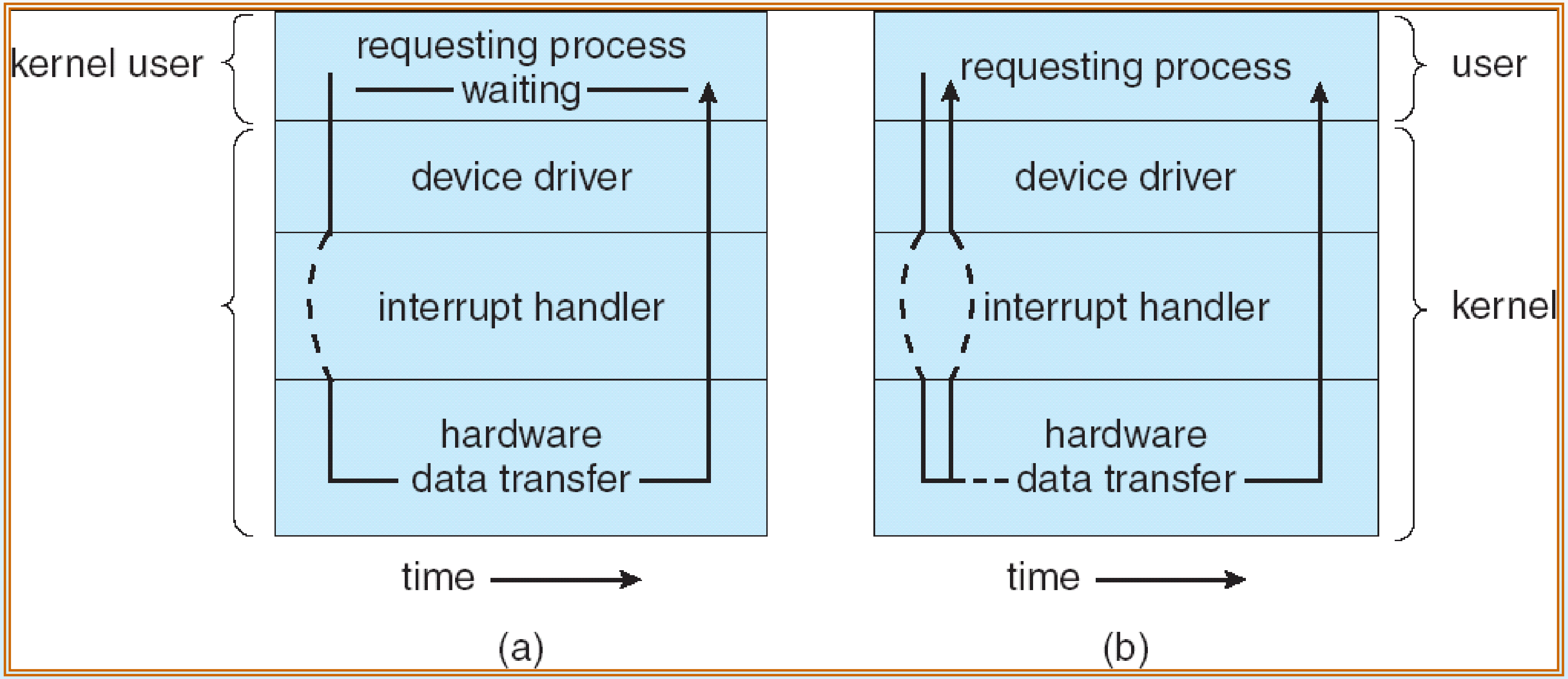
Blocking and Nonblocking I/O

- **Blocking** - process suspended (blocked) until I/O completed
  - When the process will only be waiting for one specific event.
  - Such as a disk, tape, or keyboard read by an application program.
  - **In Unix, when read file data using algorithm read()，采用 Blocking 方式。**
  - 进程必须等待某一个时间发生后才能继续执行；
  - 如登录某个系统，系统需要等待用户输入用户名、密码等信息才能继续执行；

Blocking and Nonblocking I/O

- **Nonblocking** - I/O call returns as much as available
  - **non-blocking I/O is useful** when I/O may come from more than one source and the order of the I/O arrival is not predetermined；
  - **当进程或线程同时处理多个I/O时，有些I/O可以不需要等待其完成即可继续执行**， 例如：
    - A user interface that receives keyboard and mouse input while processing and displaying data on the screen.
    - A video application that reads frames from a file on disk while simultaneously decompressing and displaying the output on the display.
    - I/O-management programs, such as a copy command that copies data between I/O devices.
  - One way an application writer can overlap execution with I/O is to write a multithreaded application. Some threads can perform blocking system calls, while others continue executing.(多线程环境下，有的线程采用**blocking I/O**，其它线程可继续执行，整体上看是非阻塞方式)；
  - **In Unix, when pre-read file data using algorithm reada() (readahead)，采用Nonblocking方式。**

Two I/O Methods—Synchronous vs. Asynchronous



Synchronous

Blocking I/O

Asynchronous

Nonblocking I/O

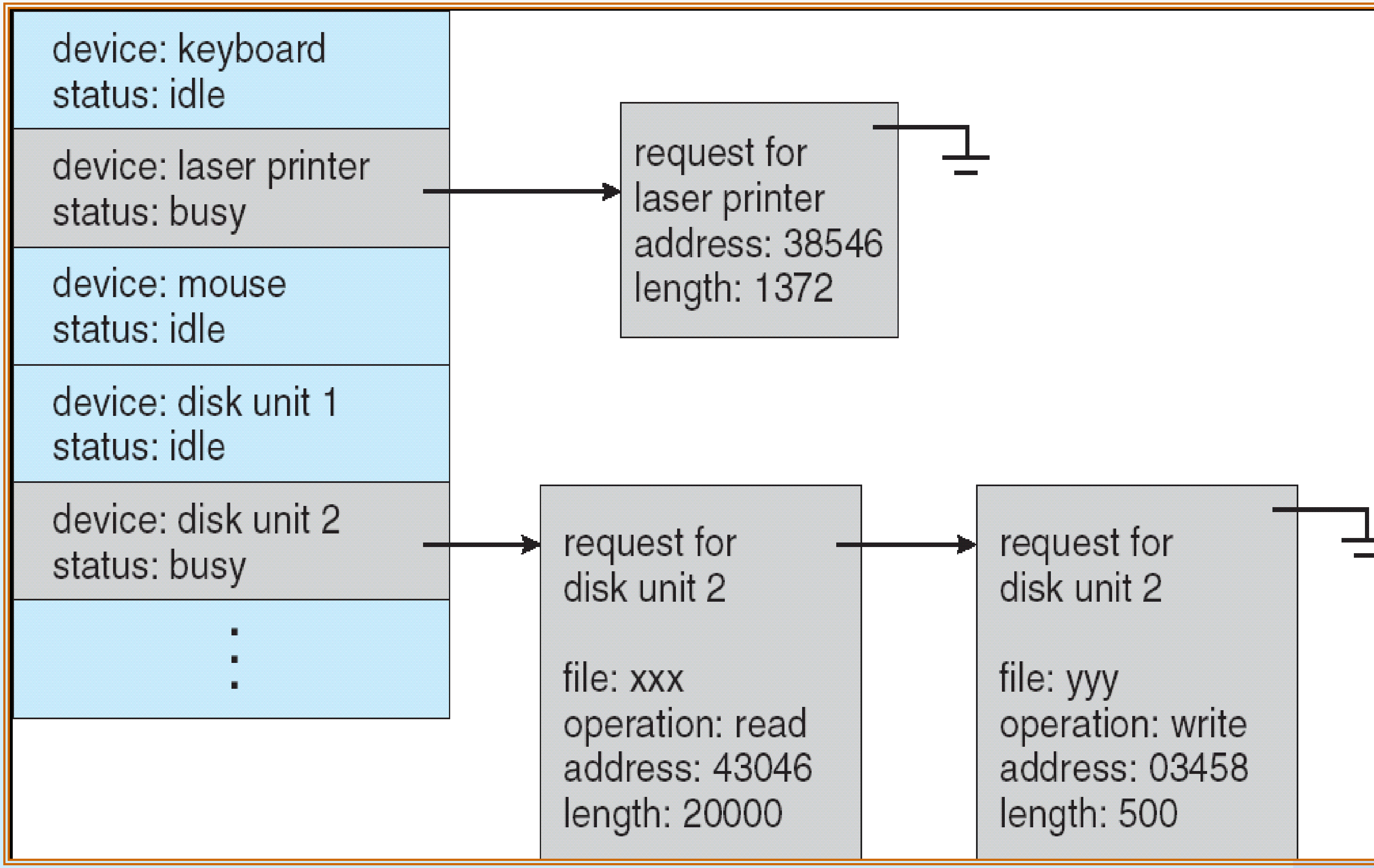
13.4 Kernel I/O Subsystem

- Kernels provide many services related to I/O
  - **Scheduling**
  - **Buffering**
  - **Caching**
  - **Spooling and Device Reservation**
  - **Error Handling**
  - **I/O Protection**

13.4.1 I/O Scheduling

- **Scheduling**
  - To schedule a set of I/O requests means to determine a good order in which to execute them；
  - Scheduling can improve overall system performance, share device access fairly among processes, and can reduce the average waiting time for I/O to complete；
  - Some I/O request ordering via per-device queue
  - Generally, scheduling is to improve the overall system efficiency and the average response time
  - Some OSs try fairness
  - 对于大部分的设备独占设备，一般采用**Non-preemptive+FCFS**调度算法
  - Disk scheduling (FCFS,SSTF,SCAN,CSCAN,LOOK,CLOOK)

Device-status Table



13.4.2 Buffering

- A **buffer** is a memory area that stores data while they are transferred between two devices or between a device and an application.
- **Buffering** - store data in memory while transferring between devices
- Why buffering?

Why buffering

- **To cope with device speed mismatch**

  (between the producer and consumer of a data stream)

- **To cope with device transfer size mismatch**

- **To maintain "copy semantics"**
  - application buffer & kernel buffer
  - e.g. write system call

Why buffering (Cont.)

- **To cope with device speed mismatch** (between the producer and consumer of a data stream)
  - for example, a file is being received via modern for storage on the hard disk.
  - a buffer is created in main memory to accumulate the bytes received from the modem.
  - When all entire buffer of data has arrived, the buffer can be written to disk in a single operation.
  - 例如对磁盘的读写（协调进程与读写磁盘之间的速度差异）、打印机输出（协调输出进程与打印机之间的速度差异）
  - 编写C程序测试printf()所使用缓存的大小

Why buffering (Cont.)

- To cope with device **transfer size mismatch**
  - Buffers are used widely for fragmentation and reassembly of messages in computer networking.
  - At the sending side, a large message is fragmented into small network packets. The packets are sent over the network.
  - The receiving side places them in a reassembly buffer to form an image of the source data
  - Store and forward (存储转发--路由器、交换机)

Why buffering (Cont.)

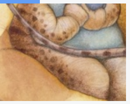
- To maintain “copy semantics” for application I/O
  - e.g. write() system call
    - 当一个进程调用**write()**将**buffer**中的数据写入磁盘，在核心将**buffer**中的数据写磁盘的过程中，进程对**buffer**中的数据进行了修改。
    - 根据“复制语义”的要求，本次写入磁盘的数据应该是在发出系统调用**write()**时**buffer**中的数据，其后的修改与本次**write()**调用无关。
    - 如果采用互斥，会降低系统的性能。
    - Suppose that an application has a buffer of data that it wishes to write to disk，It calls the write () system call
    - After the system call returns, what happens if the application changes the contents of the buffer?
    - **With copy semantics, the version of the data written to disk is guaranteed to be the version at the time of the application system call, independent of any subsequent changes in the application's buffer.**

Why buffering (Cont.)

- **To maintain "copy semantics" for application I/O**
  - **Several ways for the operating system to guarantee copy semantics**
    - **Application buffer & kernel buffer**
      - The write() system call **copy the application data** into a **kernel buffer** before returning control to the application.
      - The disk write is performed from the kernel buffer, so that subsequent changes to the application buffer has no effect.
    - The same effect can be obtained more efficiently by clever use of virtual memory mapping and copy-on-write (COW) page protection.（当用作buffer的页面被修改后，复制出一个新的页面，原来的页面用于write()，新的页面用于修改）

单选题 1分

设置



此题未设置答案，请点击右侧设置按钮

在系统内存中设置磁盘缓冲区的主要目的是（）。

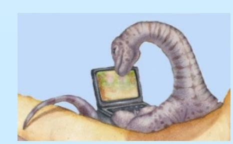
A 减少磁盘I/O的次数

B 减少平均寻道时间

C 提高磁盘数据可靠性

D 实现设备无关性

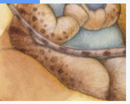
提交



单选题 1分

设置

此题未设置答案，请点击右侧设置按钮



设系统缓冲区和用户缓冲区均采用单缓冲区。从外设读入一个数据块到系统缓冲区的时间是100，从系统缓冲区读入1个数据块大用户工作区的时间为5，对用户工作区中的1个数据块进行分析的时间为90。进程从外设读入并分析两个数据块的最短时间是（）。

A 200

B 295

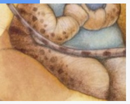
C 300

D 390

提交

单选题 1分

设置



此题未设置答案，请点击右侧设置按钮

某文件占10个磁盘块，现要把该文件磁盘块逐个读入主存缓冲区，并送用户区进行分析。假设一个缓冲区与一个磁盘块大小相同，把一个磁盘块读入缓冲区的时间为100μs，将缓冲区的数据传送到用户区的时间是50μs，CPU对一块数据进行分析的时间是50μs。在单缓冲区和双缓冲区结构下，读入并分析该文件的时间分别是（）。

A 1500μs、1000μs

B 1550μs、1100μs

C 1550μs、1550μs

D 2000μs、2000μs

提交

13.4.3 Caching

- **Caching** - fast memory holding copy of data, access to the cached copy is more efficient than access to the original one.
  - Always just a copy
  - Key to performance
- The difference between a **buffer** and a **cache**
  - **A buffer may hold the only existing copy of a data item,**
  - **Whereas a cache, by definition, just holds a copy on faster storage of an item that resides elsewhere.**
- Caching and buffering are distinct functions, but some times a region of memory can be used for both purposes.
  - (Disk Cache in Unix) The operating system uses buffers in main memory to hold disk data.
  - These buffers are also used as a cache, to improve file I/O efficiency for files that are shared by applications or that are being written and reread rapidly.

13.4.4 Spooling and Device Reservation

- 这两种技术的目的：
  - 解决独占设备的并发访问问题，以提高设备的利用率

Spooling and Device Reservation

- **SPOOLing** - hold output for a device
  - A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams
  - Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixed together.
  - **The operating system solves this problem by *intercepting* all output to the printer.**
  - Each application's output is spooled to a separate ***disk file***.
  - When an application finishes printing, the spooling system queues the corresponding spool file for output to the printer.
  - The spooling system copies the queued spool files to the printer one at a time.

SPOOLing技术

- **SPOOling**
  - **Simultaneous Peripheral Operations On Line**
  - 外部设备联机并行操作
  - 又称为假脱机

- *脱机输入*是利用专门的外围控制机将低速**I/O**设备上的数据预先输入到磁盘上，然后主机从磁盘上直接读取输入数据；
- *脱机输出*是主机先将输出数据写入到磁盘上，然后利用专门的外围控制机将磁盘上的数据在低速**I/O**设备上输出；
- 脱机**I/O**的采用提高了主机的输入输出速度；

- ***SPOOLing技术***利用一台可共享的、高速大容量的块设备（通常是硬盘）来模拟独享设备的操作，使一台独享设备变为多台可并行使用的虚拟设备，即把独享设备变为逻辑上的共享设备；
  - 类似于分时技术将一个**CPU**映射为多个**CPU**
- 给人的感觉就像是系统具有速度非常高的**I/O**设备；（如**Printer**）

SPOOLing技术

- **SPOOLing**技术利用输入进程与输出进程模拟脱机**I/O**中的专用的**I/O**控制机；
- 因此将**SPOOLing**称为联机情况下实现的外围操作；也称为假脱机操作；
- 采用**SPOOLing**技术，
  - 当用户提交一个文档给打印机时，系统为该打印请求在磁盘上创建了一个文件，然后将欲打印的文档内容写入该文件中
  - 同时在系统的打印队列中建立一张打印表
  - 系统依次将打印对列中的打印请求提交打印机打印
  - 可以理解为：基于**SPOOLing**技术，**OS**将物理打印机虚拟为磁盘上的一个文件；

SPOOLing系统的组成

- 输入井与输出井
  - 在磁盘上开辟的两块存储空间；
  - 输入井模拟脱机输入时的磁盘，收容输入数据；
  - 输出井模拟脱机输出时的磁盘，收容输出数据；
- 输入缓冲区与输出缓冲区（**buffer**）
  - 在内存中开辟的两个缓冲区
  - 输入缓冲区用于暂存输入设备输入的数据，然后传送到输入井；
  - 输出缓冲区用于暂存从输出井来的数据，然后传送给输出设备；
- 输入进程与输出进程
  - 输入进程将用户要求的数据从输入设备通过输入缓冲区送到输入井；当**CPU**需要输入数据时，直接从输入井中读入；
  - 输出进程将用户需要输出的数据送入输出井；当输出设备空闲时，从输出井读出数据，通过输出缓冲区送入输出设备上；
- 请求输出队列
  - 系统为每个请求输出的进程建立一张请求输出表；若干张请求输出表形成一个请求输出队列；当输出设备空闲时，按该队列的顺序依次输出；

SPOOLing工作过程举例
以共享打印机为例

- 当用户请求输出时，**SPOOLing**系统截获该请求，然后并不将真正的打印机分配给该用户进程，而只是为它做了两件事：

  (1) 由输出进程在输出井中为之申请一空闲存储空间，并将打印的数据写入其中；

  (2) 输出进程再为用户申请一张空白的用户请求打印表，并将用户的打印要求填入其中，再将该表挂接到请求打印队列中；
- 如果还有打印请求，**SPOOLing**系统仍然截获该请求，同样为它做上述两件事；
- 当打印机空闲时，输出进程将从打印队列中取出队首的一张请求打印表，根据表中的要求，将要打印的数据从输出井中传送到输出缓冲区，再由打印机打印；

  重复该过程，直至打印队列为空；输出进程将自己阻塞，直至新的打印请求将其唤醒；

SPOOLing系统的特点

- 采用**SPOOLing**技术管理打印机，给用户的感觉就像是每个用户独占了一台速度很高的打印机—虚拟打印机；否则用户进程将依次等待低速的打印机打印数据，降低了进程的推进速度；
- 因此我们说利用**SPOOLing**技术将一台物理设备改造成多台虚拟设备；

- 特点
  - 提高了**I/O**的速度；
  - 将独占设备改造成共享设备；
  - 实现了虚拟设备；

Spooling and Device Reservation

- **Spooling** is **one way** operating systems can coordinate concurrent output.
- **Device reservation** is **another way** to deal with concurrent device access;
- **Device reservation** - provides exclusive access to a device by enabling a process **to allocate an idle device** and to **deallocate that device when it is no longer needed.**
  - System calls for allocation and deallocation devices
  - Watch out for deadlock
  - e.g. tapes, printers.

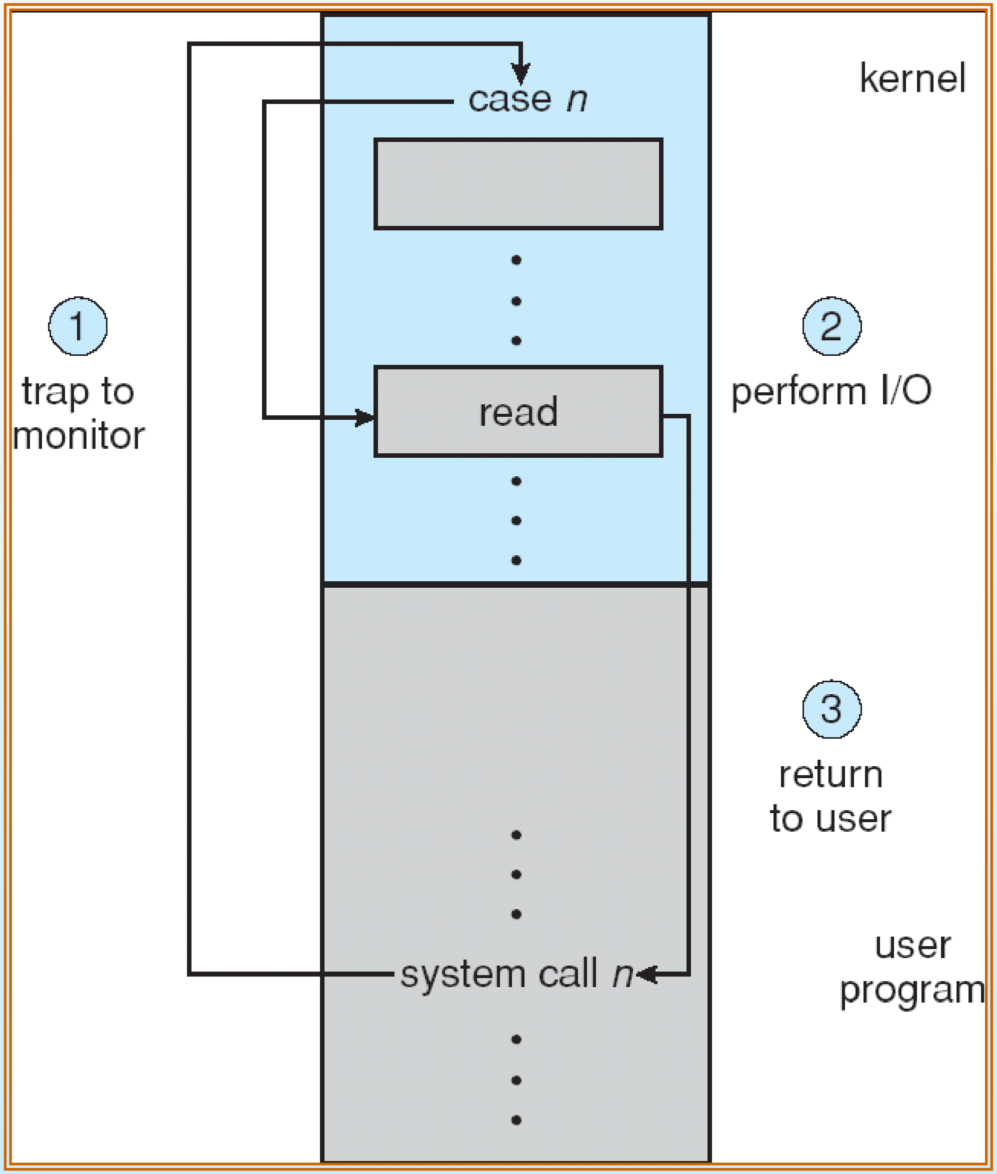
13.4.5 Error Handling

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

13.4.6 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**
  - ***<u>I/O must be performed via system calls</u>***
    - Memory-mapped and I/O port memory locations must be protected too

Use of a System Call to Perform I/O

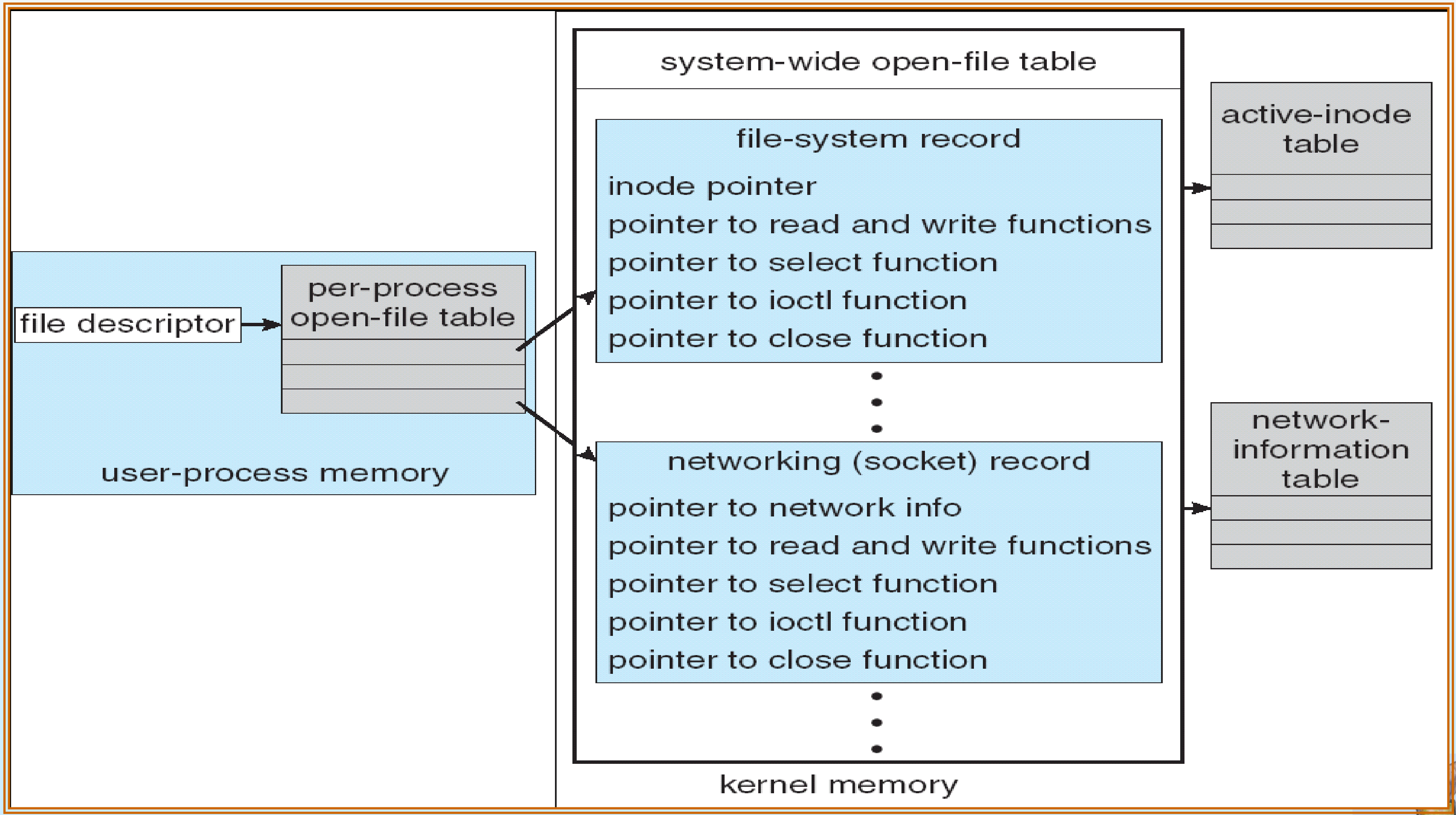


13.4.7 Kernel Data Structures

- Kernel keeps state information 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

UNIX I/O Kernel Structure

- OS将设备当做文件来看待；
- 因此系统对I/O设备的处理方式同文件的处理方式；



13.7 Performance

- **I/O is a major factor in system performance**:
  - Heavy demands on CPU to execute device driver, kernel I/O code；
  - Context switches due to interrupts (schedule processes fairly and efficiently as processes block and unblock)
    - 进程访问I/O设备时，致使进程频繁由执行进入阻塞状态，以及由阻塞进入就绪；
    - 进程访问I/O设备时，核心频繁响应中断并处理中断；
    - 导致上下文切换比较频繁；
  - Data copying (between controller and physical memory, and between kernel buffers and application data space, can **loads down** the memory bus)
  - Network traffic especially stressful (causes a high context-switch rate)

Improving Performance

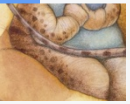
- Reduce the number of context switches.
- Reduce the number of times that data must be copied in memory while passing between device and application.
- Reduce the frequency of interrupts by using large transfers, smart controllers, and polling (if busy waiting can be minimized).
- Increase concurrency by using DMA-knowledgeable controllers or channels to offload simple data copying from the CPU.
- Move processing primitives into hardware, to allow their operation in device controllers to be concurrent with CPU and bus operation.
- Balance CPU, memory subsystem, bus, and I/O performance, because an overload in any one area will cause idleness in others.

设备独立性

- 概念
  - 应用程序独立于具体使用的物理设备
- 逻辑设备与物理设备
  - 在应用程序中使用逻辑设备名称来请求使用某类设备；在系统的实际执行时，使用物理设备名称；
  - 系统完成从逻辑设备到物理设备的映射－**LUT**
- 优点
  - 设备分配时的灵活性；
  - 易于实现**I/O**重定向(借助于**LUT**)；

单选题 1分

设置



此题未设置答案，请点击右侧设置按钮

程序员利用系统调用打开I/O设备时，通常使用的设备标识是（）。

- A 逻辑设备名
- B 物理设备名
- C 主设备号
- D 从设备号

提交

课后复习题

- P526: 3,6
- 进一步了解 P526：4，9，11
- Discussion

  1、设备驱动程序（device driver）

  2. Buffer、cache之概念以及引入它们的原因

  3、I/O is a major factor in system performance，why?

  4、How to improve I/O subsystem performance?

  5、I/O设备的保护

  6、SPOOLing的概念、组成、思想；

End of Chapter 13