上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

L12. I/O系统

宋卓然

上海交通大学计算机系

songzhuoran@sjtu.edu.cn

饮 水 思 源 • 爱 国 荣 校

概述

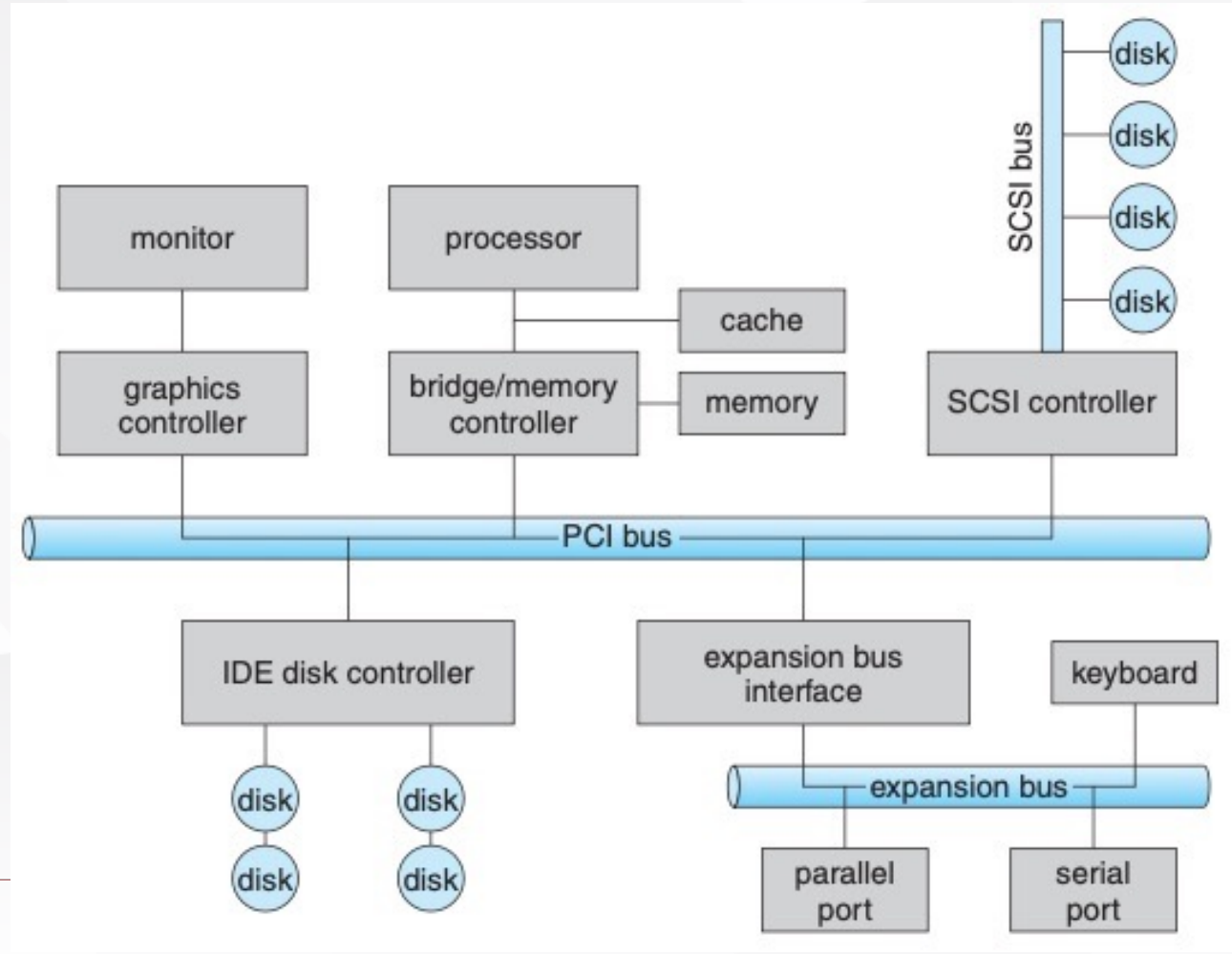
- 计算机设备控制是操作系统设计人员的主要关注之一
- I/O设备技术呈现两个突出趋势
  - 软件和硬件的接又标准化日益增长
  - I/O设备的种类也日益增多
- 为了封装各种设备的细节与特点，操作系统内核采用设备驱动程序模块
- 设备驱动程序模块为I/O子系统提供了统一的设备访问接口

I/O硬件

- 各类I/O硬件
  - 键盘
  - 屏幕
  - 鼠标
- 通过连接点或端口与计算机进行通信。如果设备共享一组通用线路，则这种连接称为总线
- 总线是一组线路和通过线路传输信息的严格定义的协议

I/O硬件

- PCI总线（PCI bus）将处理器内存子系统连到快速设备
- 扩展总线（expansion bus）连接相对较慢的设备，如键盘、USB



I/O端口

- I/O端又通常由四个寄存器组成，即状态、控制、数据输入和数据输出寄存器
  - 数据输入寄存器：被主机读出以获取数据
  - 数据输出寄存器：被主机写入以发送数据
  - 状态寄存器：包含一些主机可以读取的位，例如当前命令是否完成、 数据输入寄存器中是否有数据可以读取、是否出现设备故障等
  - 控制寄存器：可由主机写入，以便启动命令或更改设备模式。例如， 控制寄存器中的一位选择全工通信或单工通信，另一位控制启动奇偶校验检查

控制器

- 控制器可以操作端口、总线或设备的电子器件
- 处理器如何对控制器发出命令和数据以便完成I/O传输？
  - 控制器包含多个寄存器，用于数据和控制信号
  - 处理器通过读写寄存器来与控制器通信
    - 通过特殊的I/O指令，触发总线，控制I/O
    - 支持内存映射I/O，设备控制寄存器被映射到处理器的地址空间，处理器执行I/O请求是通过标准数据传输指令读写映射到物理内存的设备控制器，例如，希望打印信息到屏幕，会先将数据写入内存映射区域，图形控制器再将数据发到屏幕

主机与控制器进行交互

- 主机与控制器之间交互的完整协议可以很复杂，但基本握手概念则比较简单
- 假设采用2个位协调控制器与主机之间的生产者与消费者的关系
- 控制器通过**状态寄存器的忙位**来显示状态
  - 控制器工作忙时就置忙位
  - 可以接收下一条命令时就清忙位
- 主机通过**命令奇存器的命令就绪位**来表示意愿
  - 当主机有命令需要控制器执行时，就置命令就绪位

主机与控制器进行交互

主机与控制器之间握手的协调如下

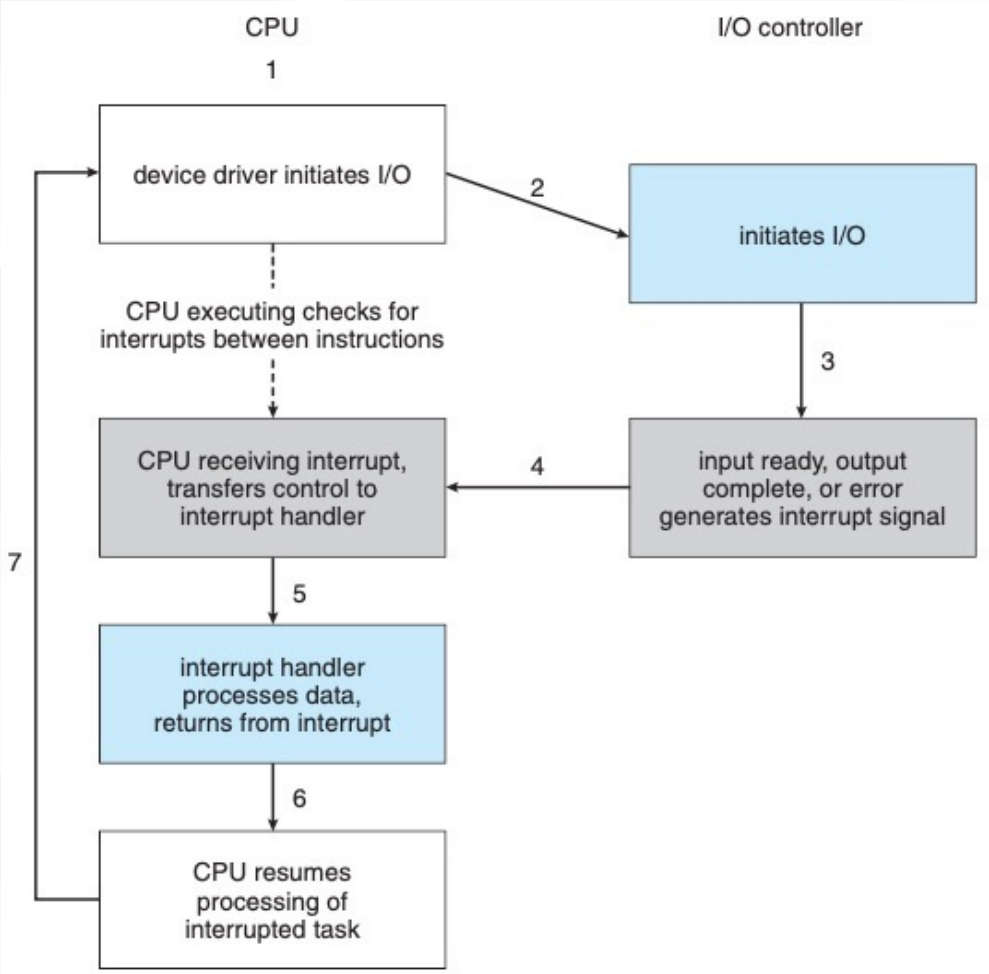
1. 主机重复读取忙位，直到该位清零
2. 主机设置命令寄存器的写位，并写出一个字节到数据输出寄存器。
3. 主机设置命令就绪位。
4. 当控制器注意到命令就绪位已设置，则设置忙位。
5. 控制器读取命令寄存器，并看到写命令。它从数据输出奇存器中读取一个字节，并向设备执行I/O操作。
6. 控制器清除命令就绪位，清除状态寄存器的故障位表示设备I/O成功，清除忙位表示完成。

主机与控制器进行交互

- 在主机与控制器之间进行握手协调时，主机处于忙等待或轮询状态
- 如果控制器和设备都比较快，响应及时，则这种方法合理；若等待时间过长，可能导致数据流失，例如来自键盘的数据
- 因此，不是要求CPU反复轮询I/O是否完成，而是让I/O设备通知CPU已完成，这种机制叫做**中断**

中断机制的工作原理

- CPU硬件有一条线，称作中断请求线（IRL）
- CPU执行完每条指令，就会检测IRL
  - 当检测到控制器已在IRL上发出一个信号，CPU保存执行状态（PCB），并跳转到内存固定位置的中断处理程序
  - 中断处理程序确定中断原因，执行必要处理，执行状态恢复，并且执行返回中断指令以便CPU回到中断前的执行状态



中断机制

- 中断机制的两个问题
  - 需要轮询所有设备，才能知道哪个引起了中断，十分低效
    - 中断向量
  - 假设有多个中断到来，无法知道应该先处理哪个中断
    - 采用多级中断

中断机制

- 中断机制通过一个地址，来从集合中选择特定的中断处理程序
  - 地址称为**中断向量**表中的偏移量
  - 这个向量包含了专门的中断处理程序的内存地址
  - 目的：不再需要搜索所有可能的中断源，以决定哪个中断需要服务，而是直接确定中断源

中断机制

- 中断机制实现了中断优先级系统
  - 延迟处理低优先级的中断，而非屏蔽所有中断
  - 允许高优先级中断抢占执行低优先级中断
  - 解决了多个中断等待的响应问题
- 中断机制也可用于处理异常
  - 除以0
  - 访问保护的、不存在的内存地址
  - 尝试执行源自用户模式的特权指令
- 触发中断的事件有一个共同特点：这些事件导致操作系统执行紧急的自包含的程序

直接内存访问

- 对于执行大量传输到设备，如磁盘，如果通过昂贵的CPU来观察状态位并按字节发送数据到控制器寄存器（称为程序控制I/O），很浪费
  - 为大量传输设计了专用处理器，称为直接内存访问（direct-memory access，DMA）
- 启动DMA传输时：
  - 主机将DMA命令写到内存，该命令块包含传输来源地址的指针、传输目标地址的指针、传输的字节数
  - CPU将这个命令块写到DMA控制器
  - DMA控制器继续操作内存总线，将地址放到总线，在没有主CPU的帮助下进行数据传输

直接内存访问

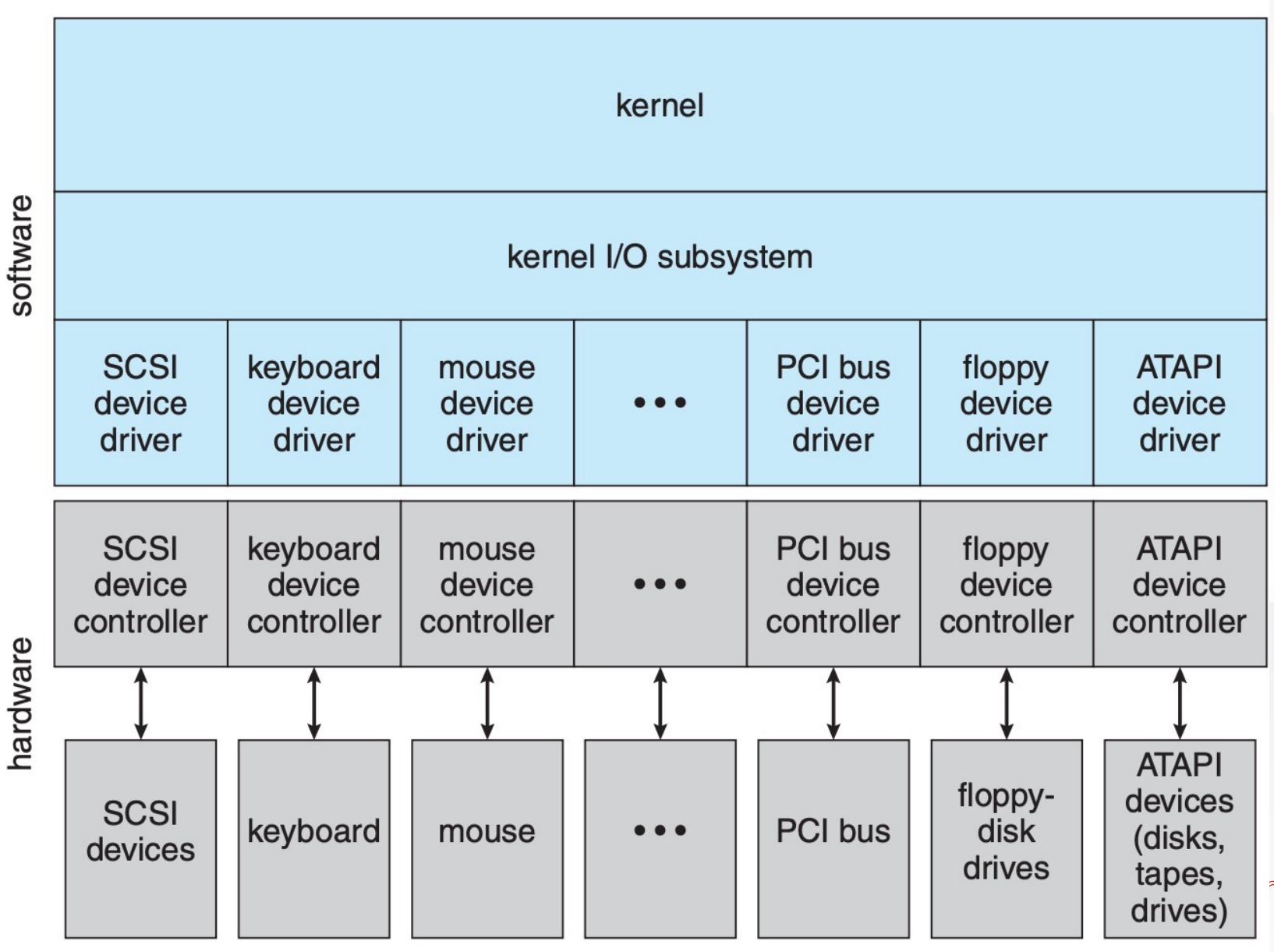
- DMA控制器与设备控制器之间的握手，通过一对称为DMA请求和DMA确认的线路来进行
- 当有数据需要传输时，设备控制器发送信号到**DMA请求线路**，这个信号使DMA控制器占用内存总线，发送所需地址到内存地址总线，并发送信号到**DMA确认线路**。当设备控制器收到DMA确认信号时，它就传输数据到内存，清除DMA请求信号

虽然DMA取得内存总线控制权后CPU无法使用内存，但依然可以使用cache，从整体性能上有所提升

应用程序I/O接口

- 如何用统一的标准来处理I/O设备？
  - 例如应用程序在打开磁盘文件时，不必知道它在哪个磁盘
- 抽象通用类型，每种通用类型可以通过一组标准函数（即接口（interface））来访问
- 设备差异被封装到内核模块（称为设备驱动程序）
  - 设备驱动程序可以定制以适应不同的设备
  - 向上提供一组标准接口

应用程序I/O接口


kernel
kernel I/O subsystem
software
SCSI device driver
keyboard device driver
mouse device driver
• • •
PCI bus device driver
floppy device driver
ATAPI device driver
SCSI device controller
keyboard device controller
mouse device controller
• • •
PCI bus device controller
floppy device controller
ATAPI device controller
hardware
SCSI devices
keyboard
mouse
• • •
PCI bus
floppy-disk drives
ATAPI devices (disks, tapes, drives)

设备驱动程序

- 设备驱动层是为了隐藏**设备控制器间的差异**
- 设备在不同操作系统上都有自己的设备驱动接口标准，可能带有多个设备驱动程序，例如针对Windows、Linux、Mac OS X的驱动程序

方面	差异	例子
数据传输模式	字符，块	终端，磁盘
访问方式	顺序，随机	调制解调器，光盘
传输方式	同步，异步	磁带，键盘
分享	专用，共享	磁带，键盘
设备速度	延迟，寻道时间，传输速率，操作延迟	
I/O 方向	只读，只写，读写	光盘，图形控制器，磁盘

块与字符设备

- 块设备接口为磁盘等基于块访问的设备而设计
  - read(), write()
  - 如果是随机访问设备，则提供seek()来指定下一个传输块
- 字符流接口为键盘、鼠标、打印机等基于字符访问的设备而设计
  - get(), put()

时钟与定时器

- 时钟与定时器主要提供三个功能
  - 获取当前时间
  - 获取经过时间
  - 设置定时器，以便在T时触发操作X
- 主要用于对时间敏感的应用程序
- 测量经过时间和触发操作的硬件称为可编程间隔定时器，它可以设置等待一定时间，然后触发中断
  - 调度程序应用它，用于时间片到达抢占进程
  - 磁盘I/O子系统采用它，定期刷新脏的缓存缓冲到磁盘
  - 网络子系统采用它，定时取消由于网络拥塞导致阻塞的操作

非阻塞与异步I/O

- 系统调用接口可以选择阻塞I/O或非阻塞I/O
  - 应用程序执行阻塞系统调用时，应用程序的执行会被挂起
  - 用户级进程可能需要使用非阻塞I/O
    - 例如接收键盘和鼠标输入，由于输入是不定期的，使用阻塞I/O显然不合理
- 非阻塞系统调用的一种替代方法是异步系统调用
  - 异步调用立即返回，无需等待I/O完成，应用程序继续执行代码。在将来I/O完成时，或通过设置应用程序地址空间内某个变量，或通过触发信号，或软件中断的方式，来通知应用程序

非阻塞与异步I/O

- 非阻塞系统调用与异步系统调用的区别
  - 非阻塞调用read()立即返回任何可用的数据，读取的数据等于或少于请求的字节数，或为零
  - 异步调用read()要求的传输会完整执行，但是完成是在将来的某个特定时间

内核I/O子系统

- 内核提供与I/O相关的许多服务
  - 调度
  - 缓冲
  - 缓存
  - 假脱机
  - 设备预留
  - 错误处理

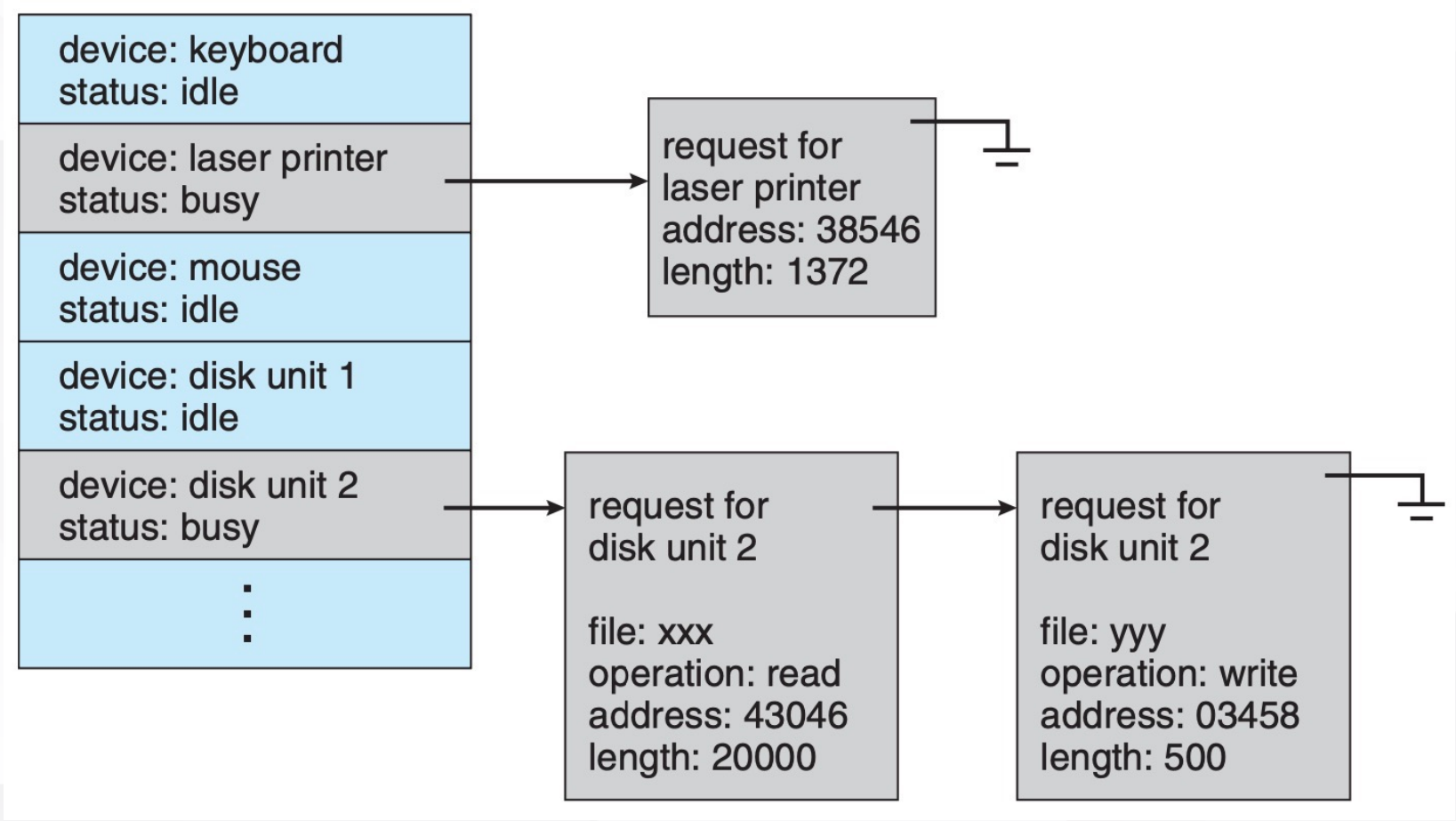
内核I/O子系统 I/O调度

- 有多个I/O请求，需要定义I/O调度方法，确定I/O请求的处理顺序
- 调度可以改善系统整体性能，可以在进程间公平共享设备访问，减少I/O完成的平均等待时间
  - 考虑一个磁盘，磁臂位于磁盘开头，三个应用程序对磁盘进行阻塞读取调用。程序1请求磁盘结束附近的块，程序2请求磁盘开始附近的块，程序3请求磁盘中间的块。操作系统按照2、3、1的顺序处理应用程序，可以减少磁臂移动距离。

内核I/O子系统 I/O调度

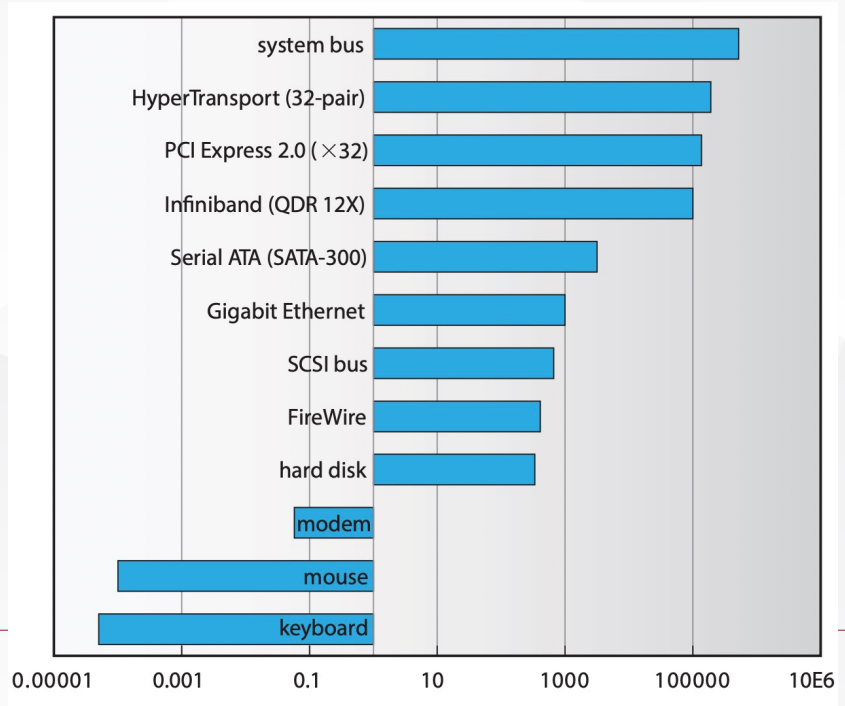
- 操作系统为每个设备维护一个请求等待队列，当应用程序发出阻塞I/O的系统调用时，该请求会被添加到队列
- I/O调度程序重新安排队列顺序，以便提高系统的总体效率
  - 之前讨论过磁盘I/O的多个调度算法，如FCFS等
- 当内核支持异步I/O时，它能够同时跟踪多个I/O请求。操作系统会将等待队列附加到设备状态表
  - 内核管理该表，其中每个条目对应每个I/O设备的类型、状态
  - 如果有多个请求，则将请求的操作、地址、长度记录下来

内核I/O子系统 I/O调度



内核I/O子系统 缓冲

- 缓冲区是一块内存区域，用于保存在两个设备间或设备与应用程序间传输的数据
- 采用缓冲的理由
  - 处理数据流的生产者与消费者之间的速度不匹配，如调制解调器与硬盘传输



内核I/O子系统 缓冲

- 缓冲区是一块内存区域，用于保存在两个设备间或设备与应用程序间传输的数据
- 采用缓冲的理由
  - 处理数据流的生产者与消费者之间的速度不匹配，如调制解调器与硬盘传输
  - 协调传输大小不一数据的设备，可以利用缓冲区重组数据

内核I/O子系统 缓存

- 缓存（cache）是保存数据副本的高速内存区域，访问缓存副本比访问原版更加有效
- 缓存与缓冲的功能不同，但一个内存区域可以当作缓存与缓冲使用。例如，为了保留复制语义和有效调度磁盘I/O，操作系统采用内存中的缓冲区来保存磁盘数据，这些缓冲区也可用于缓存，以提供文件的I/O效率，这些文件可以被多个程序共享，或者快速写入和重读。当内核收到文件I/O请求时，内核首先访问缓冲区缓存，以便查看文件区域是否在内存中，从而避免访问物理磁盘I/O

内核I/O子系统 假脱机

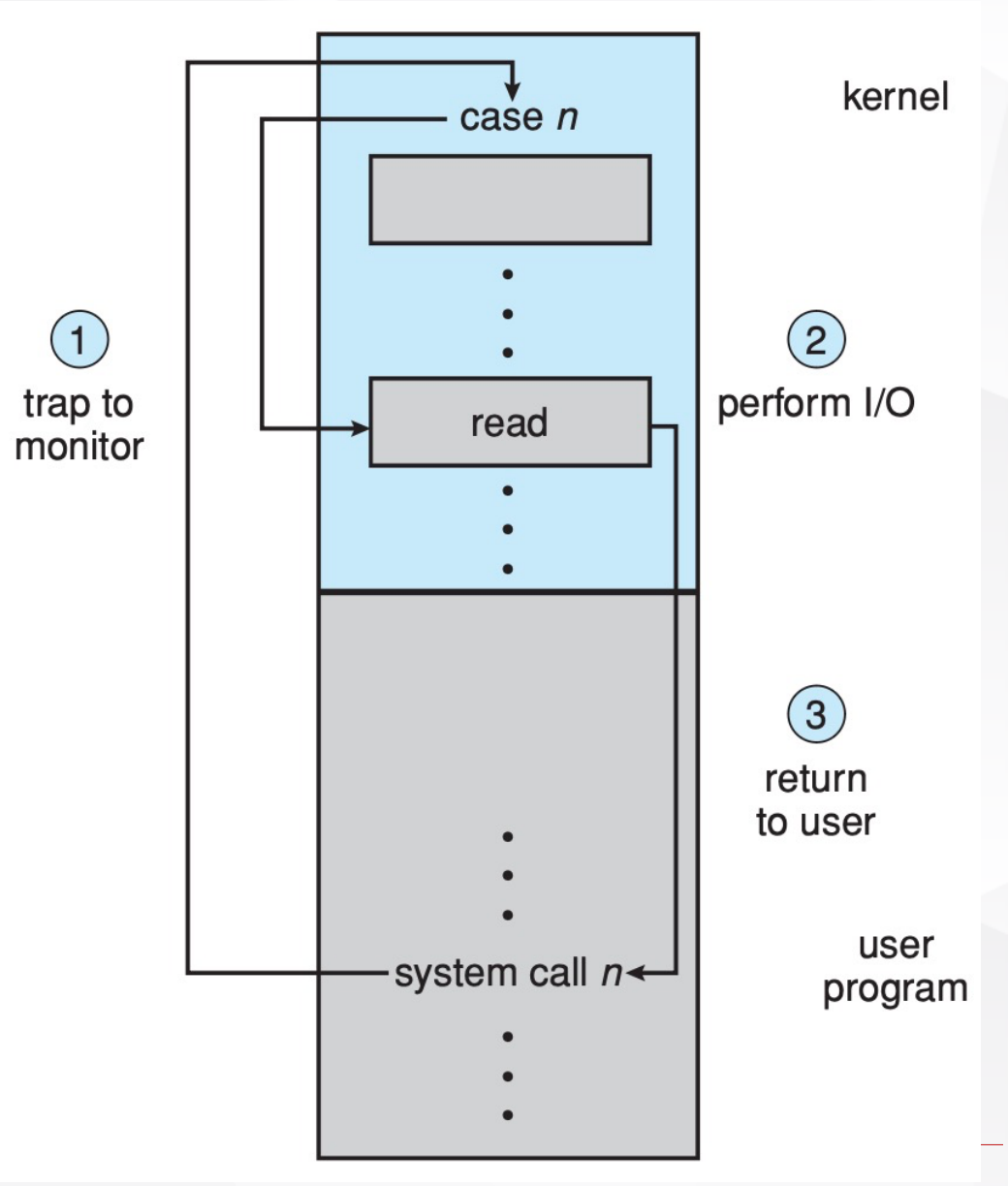
- 假脱机是保存设备输出的缓冲区，针对设备无法接收交叉数据流的情况
  - 对于打印机，虽然打印机一次只能打印一个任务，但当多个应用程序希望并发打印输出时，操作系统通过拦截所有打印输出来解决这一问题
  - 应用程序的输出先假脱机到一个单独的磁盘文件，当应用程序完成打印时，假脱机系统排序相应的假脱机文件，以便输出到打印机

内核I/O子系统 错误处理

- 设备和I/O传输的故障可能有多种原因
  - 网络超载
  - 磁盘写坏
- I/O系统调用通常返回一位调用状态信息，以表示成功或失败
  - 对于UNIX操作系统，返回错误代码约有100个，以便指出失败的大致性质，例如参数操作范围、坏指针、文件不存在
  - 有的硬件可以提供很详细的错误信息，例如SCSI协议定义了设备故障的三个等级：感应键，用于标识故障的一般性质，如硬件错误或非法请求；额外感应代码，用于表示故障类型；额外感应代码修饰词，给出详细信息，哪个硬件错误等

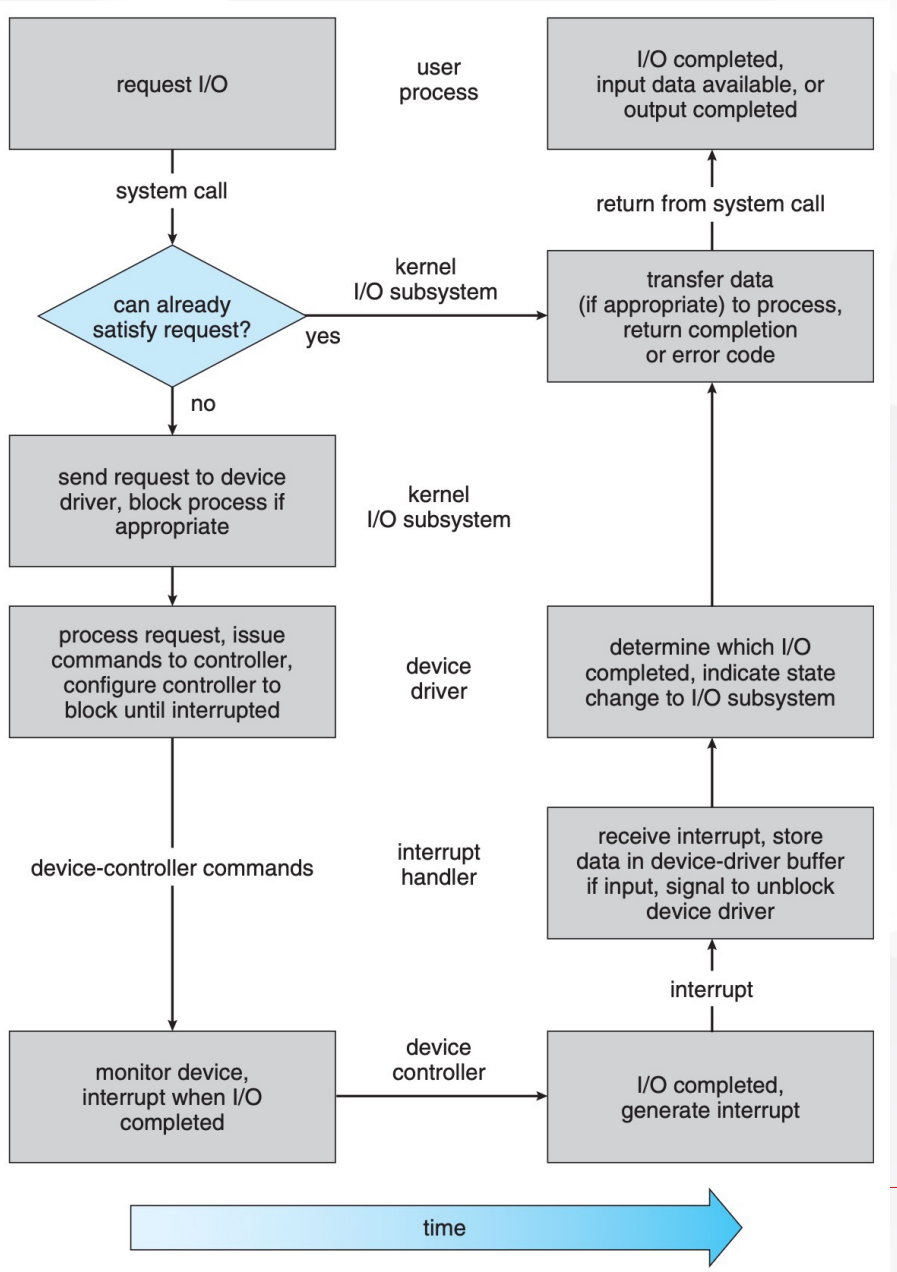
内核I/O子系统 I/O保护

- 错误与保护问题密切相关。用户程序通过试图发出非法I/O指令，可能有意或无意地中断正常系统操作。因此可以采用各种保护机制，确保系统不会发生中断
- 定义所有I/O指令为特权指令，不能由用户直接发出，而是通过操作系统来进行
- 为了进行I/O，用户程序执行系统调用，请求操作系统代表用户程序执行I/O操作，而操作系统在监控模式下检查请求是否合法，如果合法则处理请求



I/O请求的生命周期 以读文件为例

1. 针对以前已经打开文件的文件描述符，进程调用阻塞系统调用read()
2. 内核系统调用代码检查参数是否正确。对于输入，如果数据已在缓冲缓存，则数据返回到进程，并完成I/O请求
3. 否则，必须执行物理I/O请求。该进程从运行队列移到设备的等待队列，并调度I/O请求。最后，I/O子系统发送请求到设备驱动程序

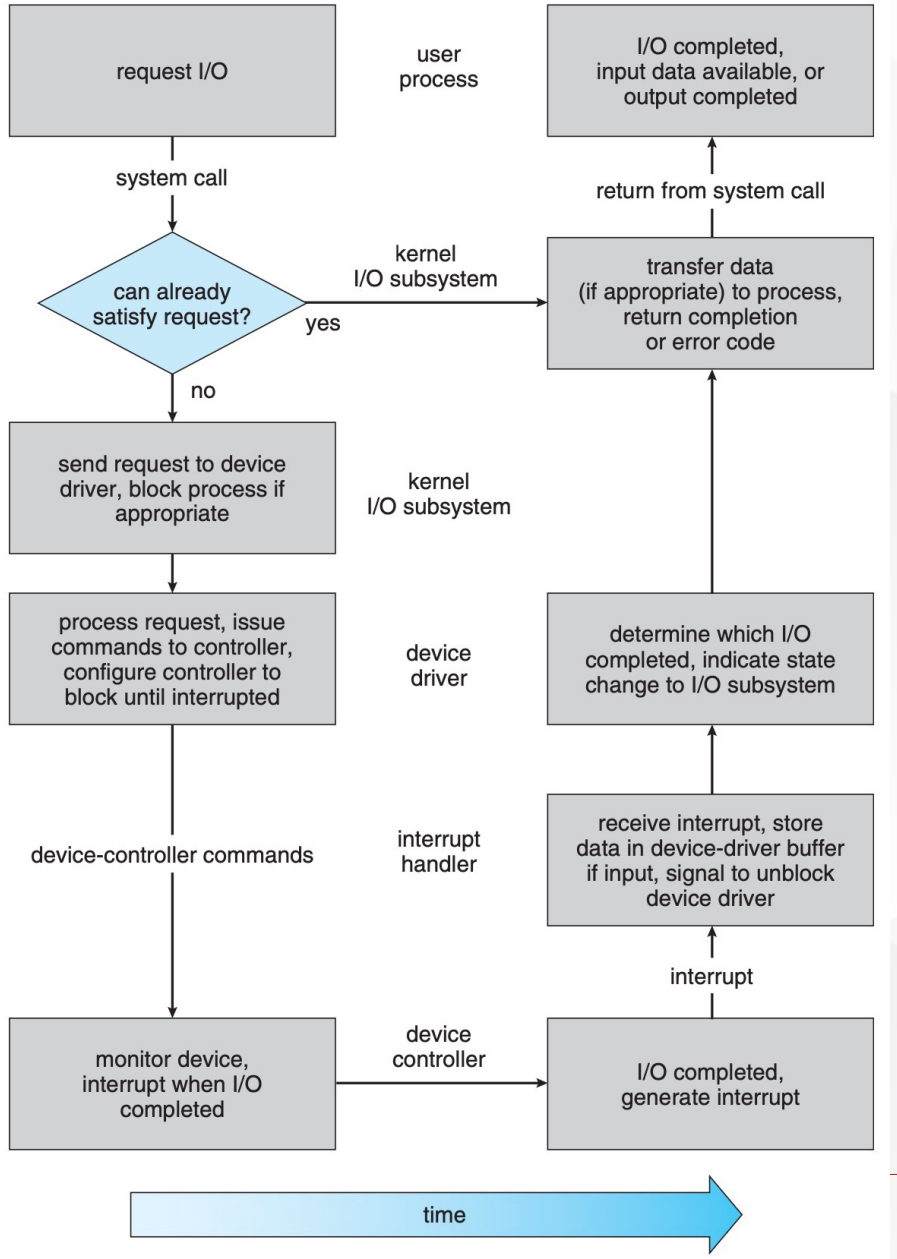


I/O请求的生命周期 以读文件为例

4．设备驱动程序分配内核缓冲区空间，来接收数据并调度I/O。最终，设备驱动程序通过写入设备控制器寄存器，对设备控制器发送命令

5. 设备控制器控制设备硬件，以便执行数据传输

6. 驱动程序可以轮询检测状态和数据，或者它可以通过DMA来传输到内核内存。假设DMA控制器管理传输，当传输完成时它会产生中断

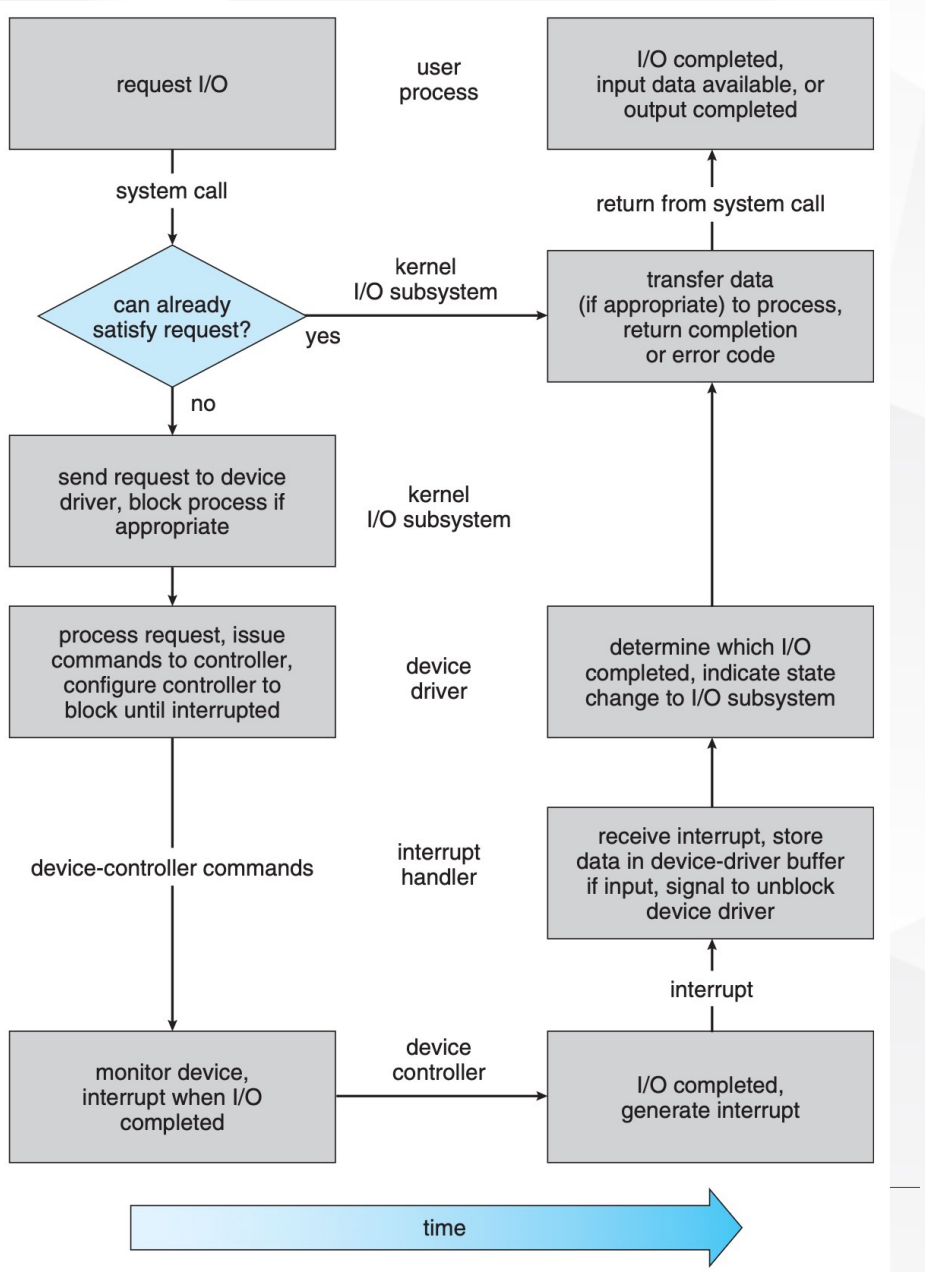


I/O请求的生命周期 以读文件为例

7. 正确的中断处理程序通过中断向量表收到中断，保存任何必要的数据，并向内核设备驱动程序发送信号通知，并从中断返回

8. 设备驱动程序接收信号，确定I/O请求是否完成，确定请求状态，并对内核I/O子系统发送信号来通知请求已经完成

9. 内核传输数据或返回代码到请求进程的地址空间，并且将进程从等待队列移到就绪



性能

- I/O是系统性能的主要因素之一
  - 需要CPU执行设备驱动程序、内核I/O代码
  - 由于中断而切换上下文，增加了CPU及其硬件缓存的负担
  - 控制器与物理内存之间的数据复制

改善I/O效率

- 改善I/O效率的方法
  - 减少上下文切换的次数
  - 减少设备和应用程序之间传递数据时的内存数据的复制次数
  - 通过大传输、智能控制器、轮询(如果忙等可以最小化)，减少中断频率
  - 通过DMA智能控制器和通道来为主CPU承担简单数据复制，增加并发
  - 将处理原语移到硬件，允许控制器操作与CPU和总线操作并发
  - 平衡CPU、内存子系统、总线和I/O的性能，因为任何一处的过载都会引起其他部分的空闲