操作系统

L23 I/O子系统（下）

胡燕
大连理工大学 软件学院

典型的I/O控制方式

通道控制IO

DMA

中断控制IO

程序控制IO

2-I/O控制方式

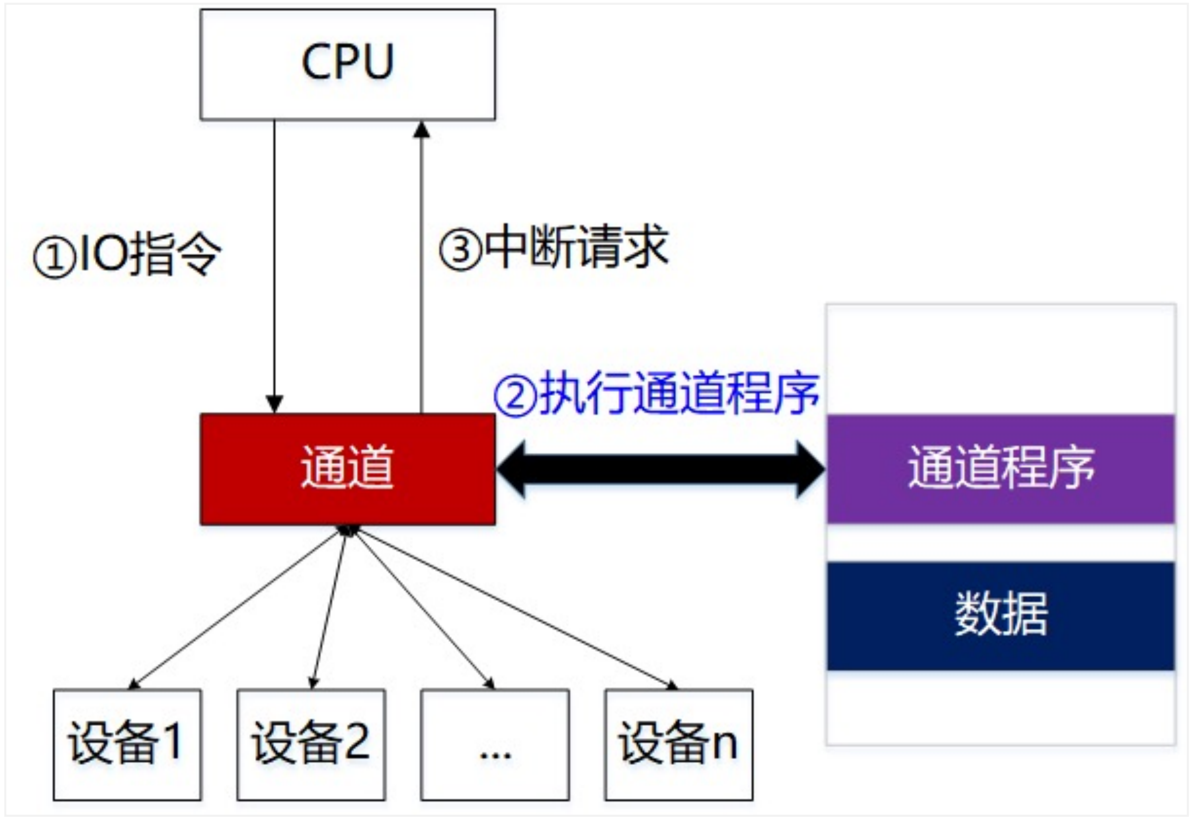
通道

大型计算机系统中采用通道处理机
- 使CPU摆脱繁重的IO处理负担
- 共享IO接口

大型计算机系统中，如果仅采用程序控制、DMA等常规IO控制方式，在进行设备管理时，会面临如下问题
- 大量外围设备的I/O工作要由CPU管理，挤占CPU支持应用程序执行的时间
- 大型计算机系统中的外围设备很多，但一般不同时工作
    - 为每个设备都配备一个接口，代价很高

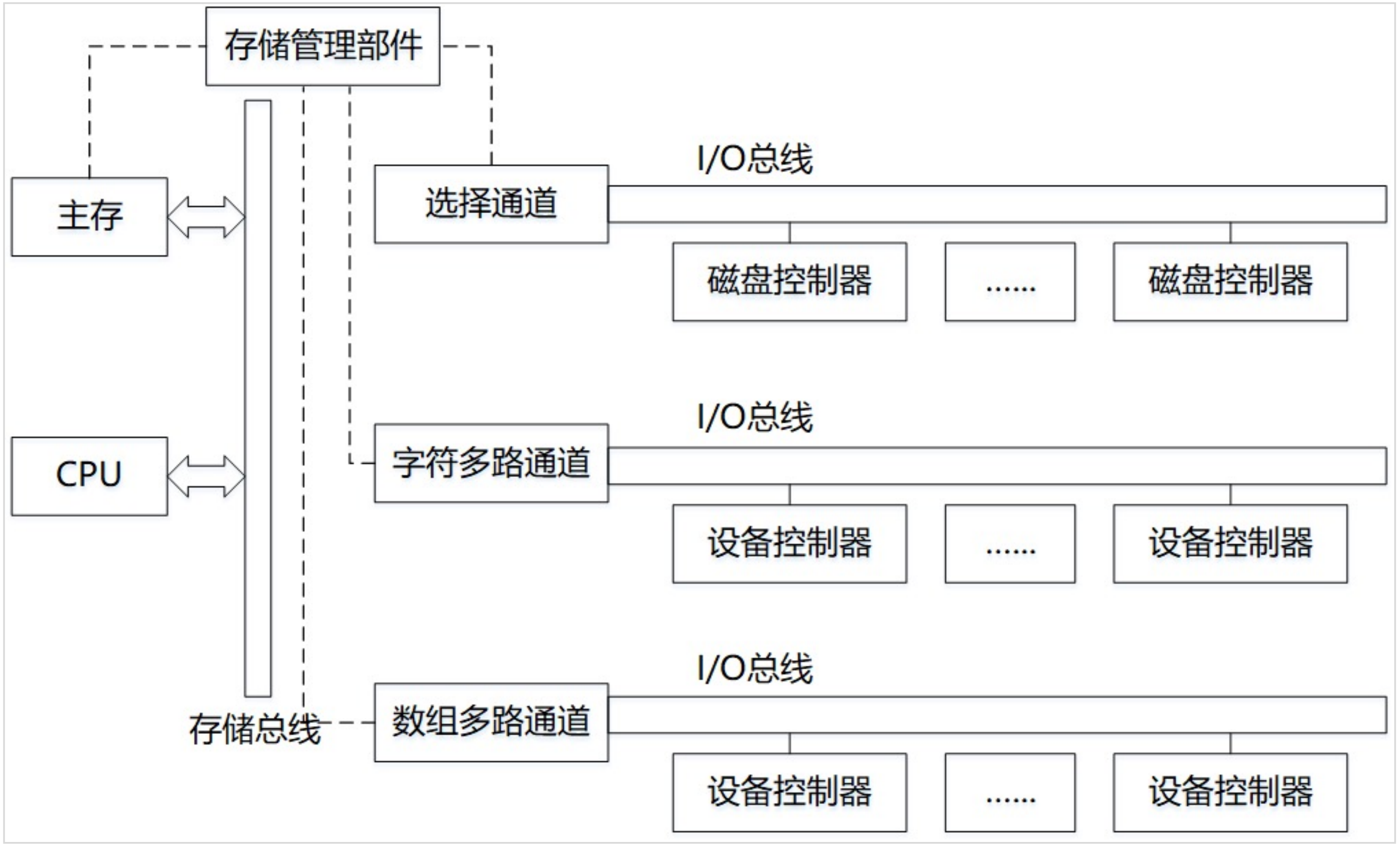
通道

Channelled IO
通道控制



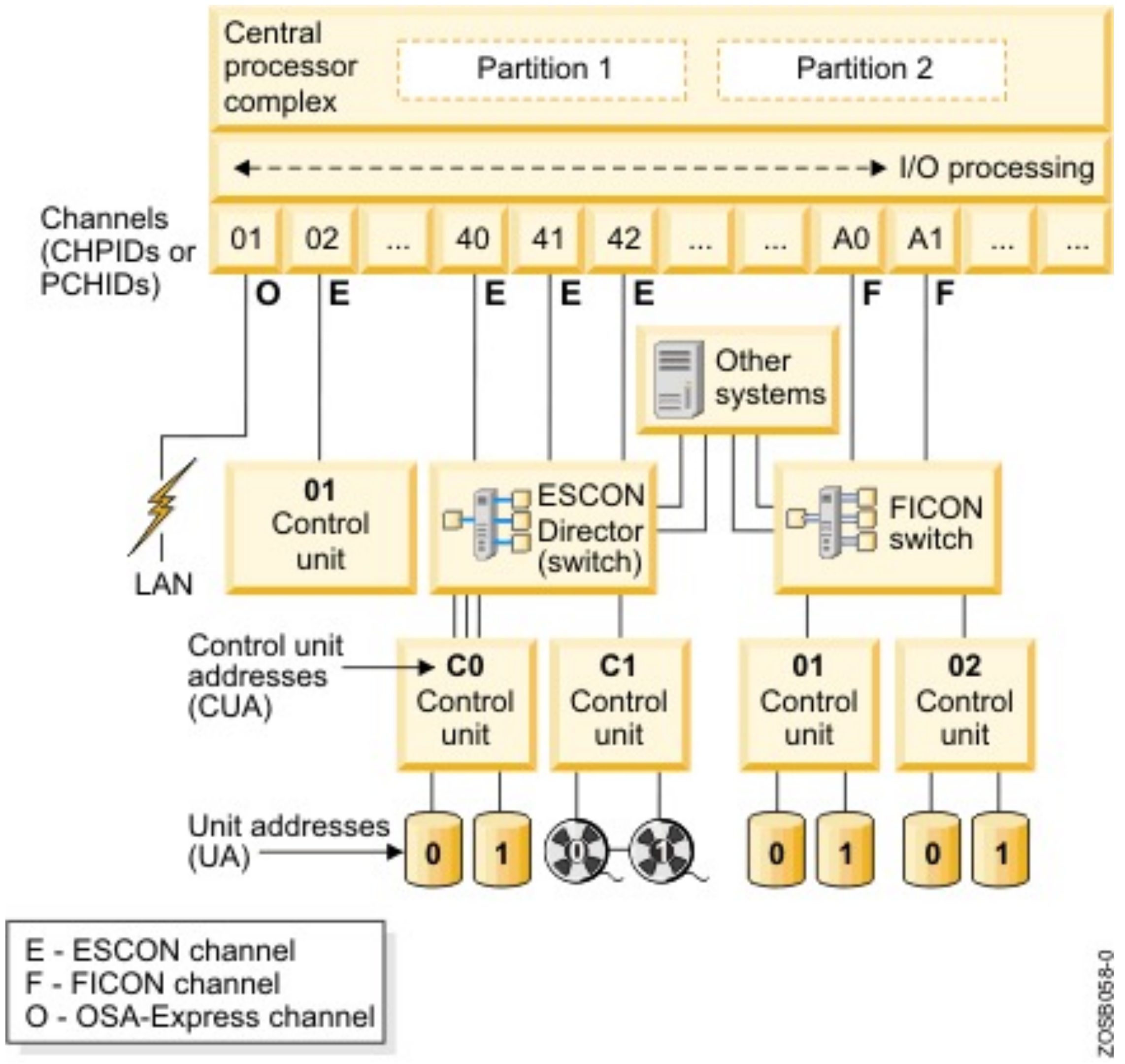
通道

具有通道的计算机系统典型结构



通道

IBM z9 io structure Demostration



E - ESCON channel
F - FICON channel
O - OSA-Express channel

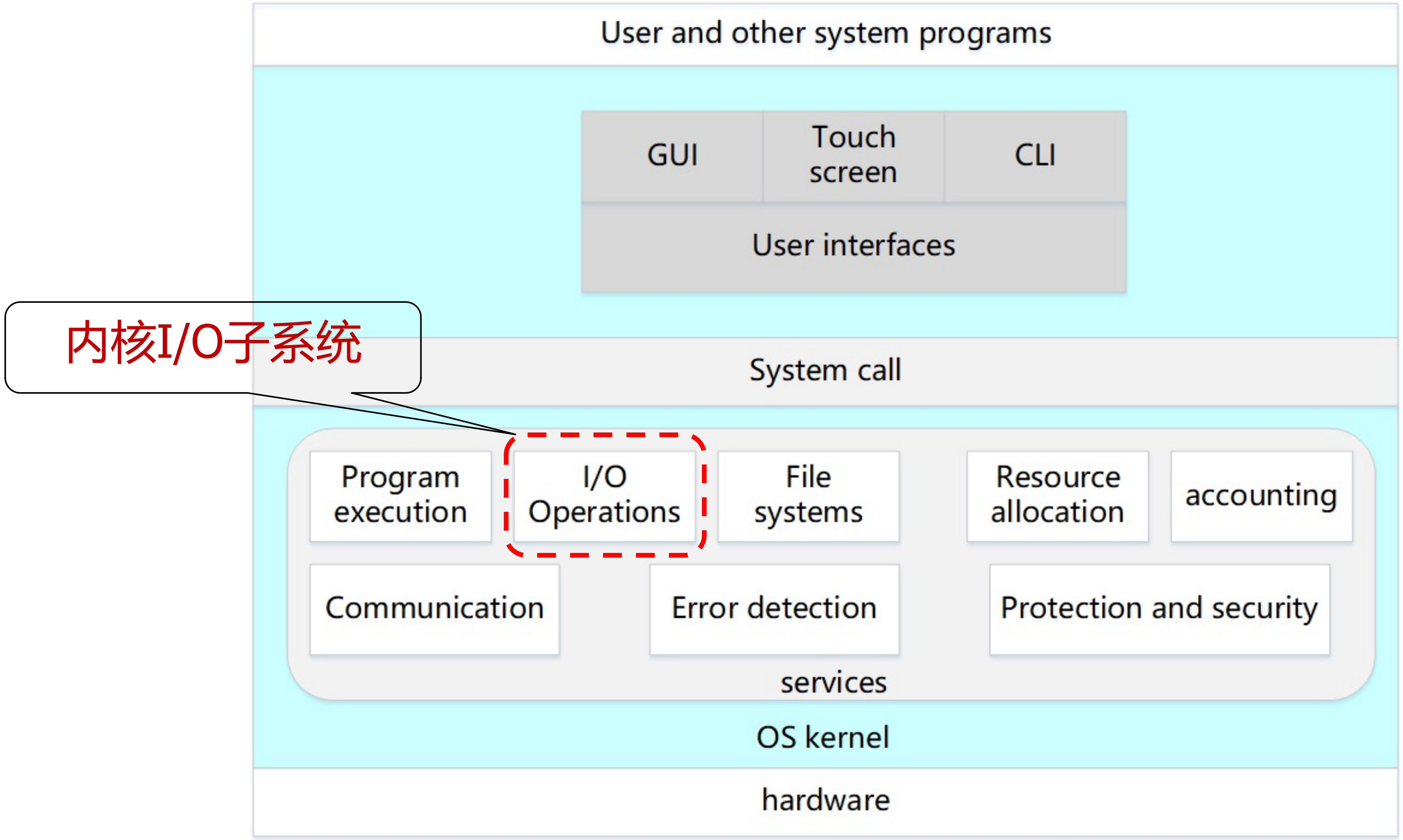
I/O软件

I/O Software

01

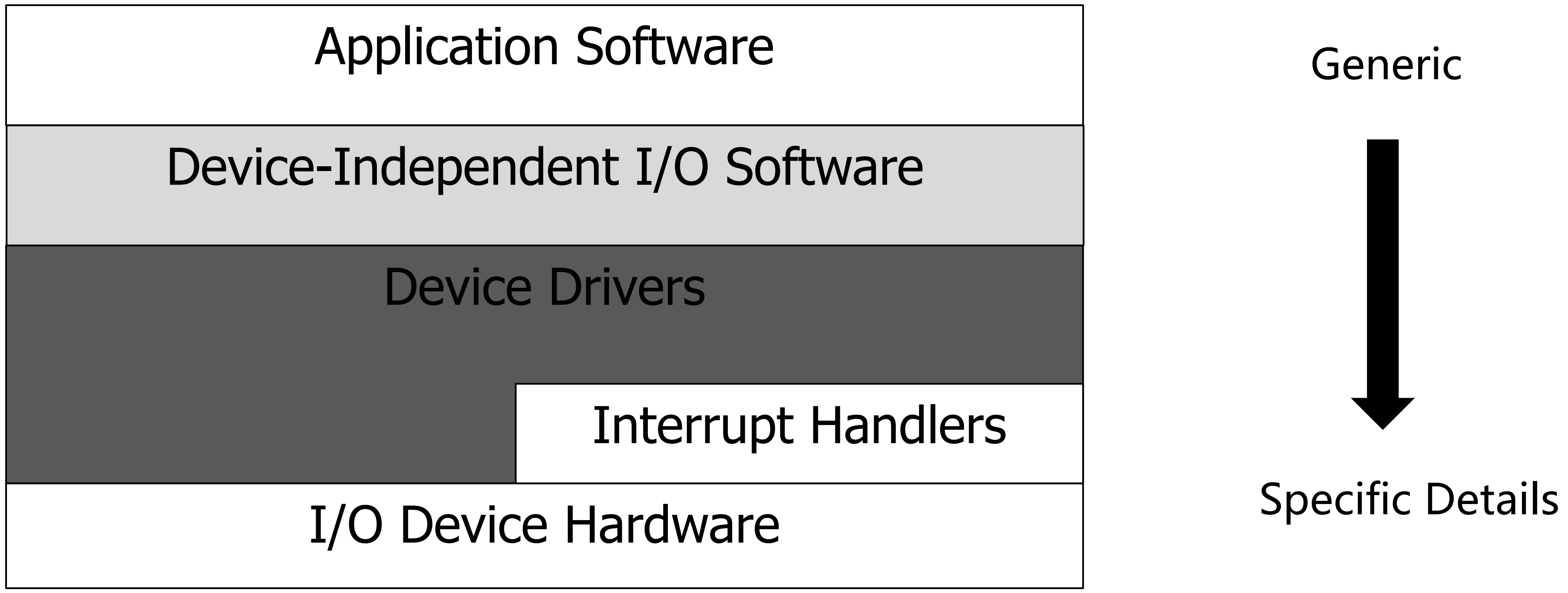
1-I/O软件

I/O软件设计层次



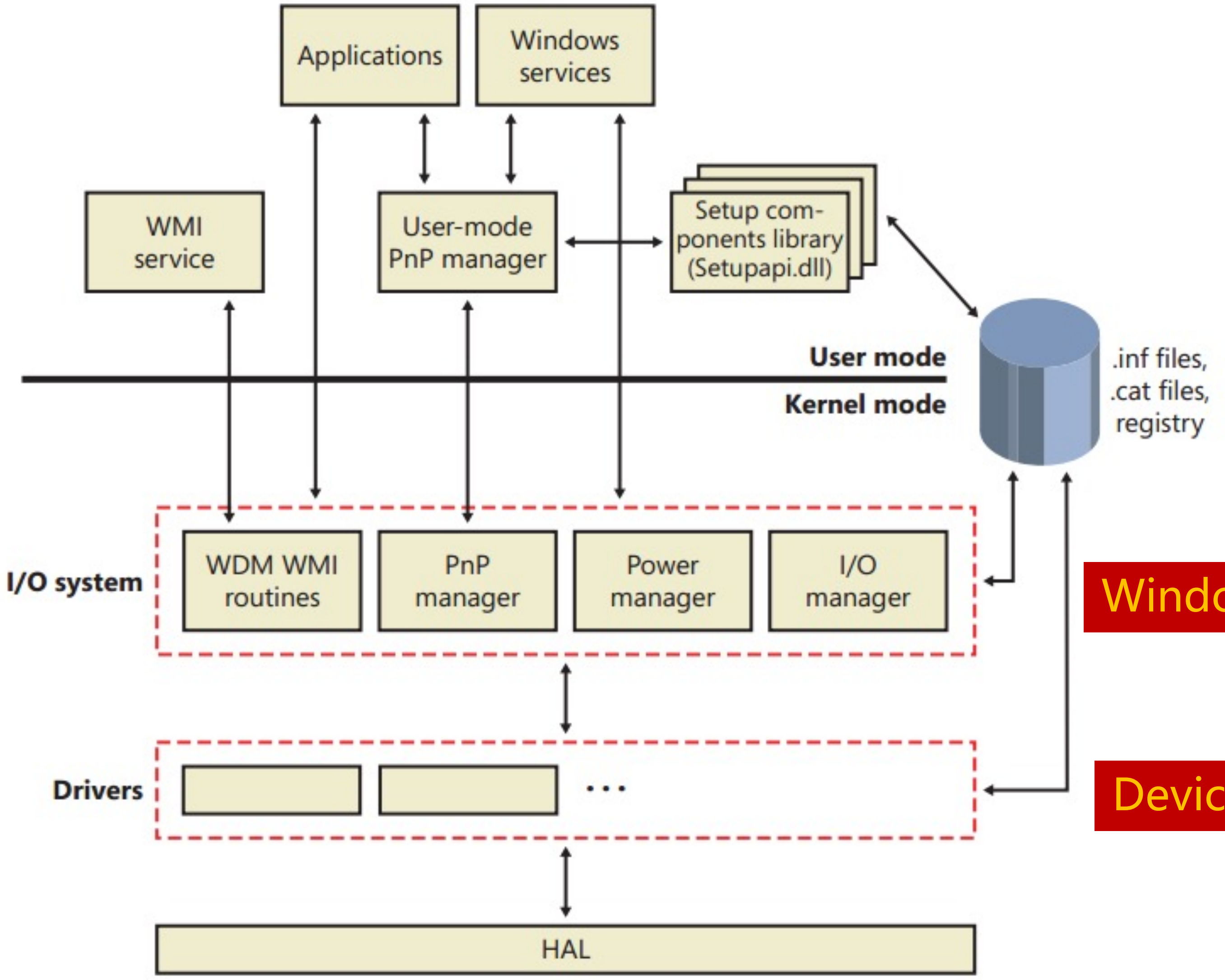
I/O软件设计层次

IO软件的设计思路：
通过层次设计，利用低层I/O软件层屏蔽硬件细节；
以低层设备驱动软件接口为基础，形成设备无关的I/O软件中间层；
通过I/O系统调用将I/O系统服务提供给应用层；



I/O软件设计层次

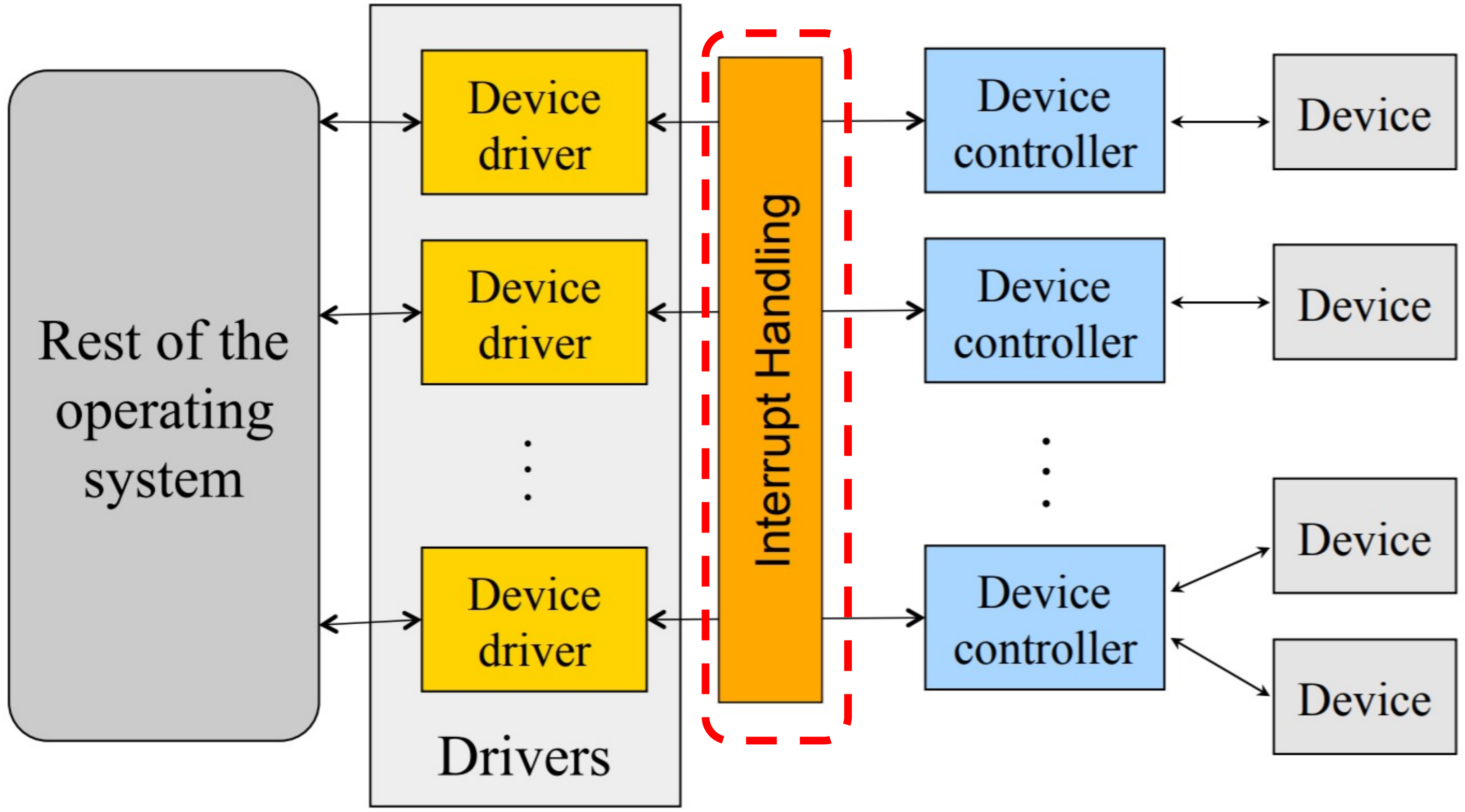
IO软件分层设计示例：Windows IO子系统



Windows I/O System

Device Driver Layer

中断处理程序（Interrupt Handler）

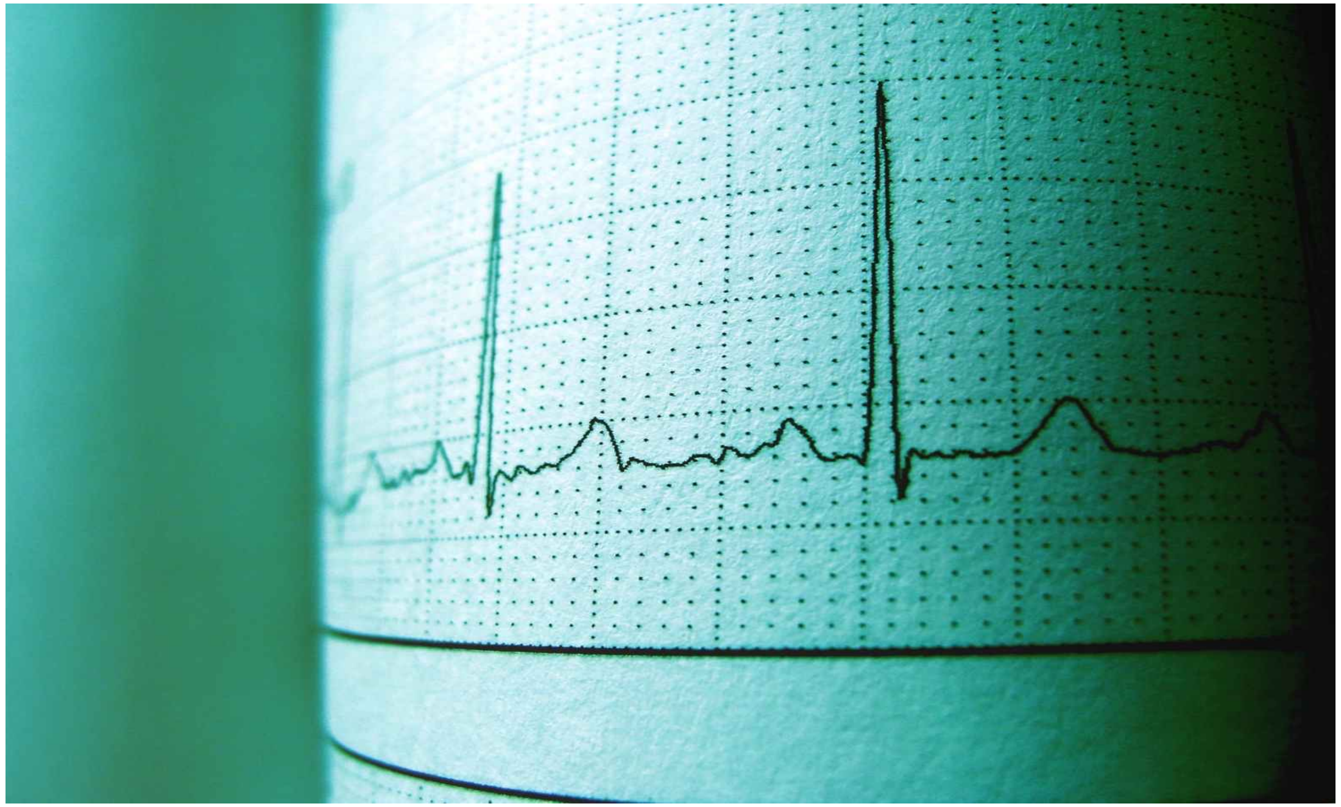


I/O设备驱动程序，需要中断处理程序作为关键基础支撑

中断处理程序

中断概念

An interrupt is a signal emitted by a device attached to a computer or from a program within the computer.

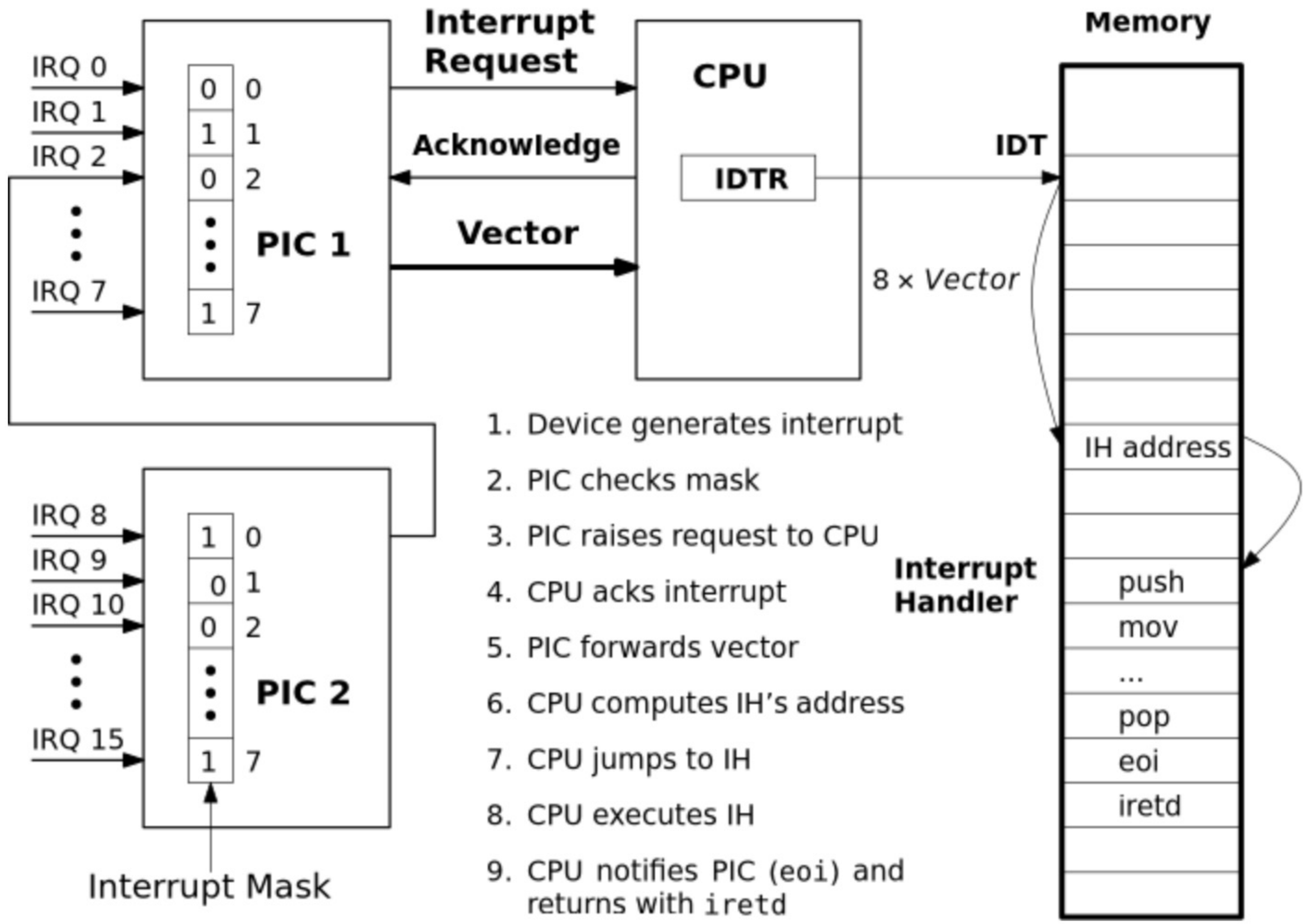


Interrupts are the heartbeats of a modern OS kernel.

中断处理程序

PC中断硬件

Example of PC interrupt handling

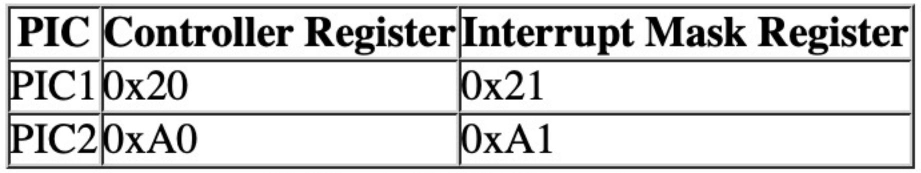


1. Device generates interrupt
2. PIC checks mask
3. PIC raises request to CPU
4. CPU acks interrupt
5. PIC forwards vector
6. CPU computes IH's address
7. CPU jumps to IH
8. CPU executes IH
9. CPU notifies PIC (`eoi`) and returns with `iretd`

1-I/O软件　中断处理程序

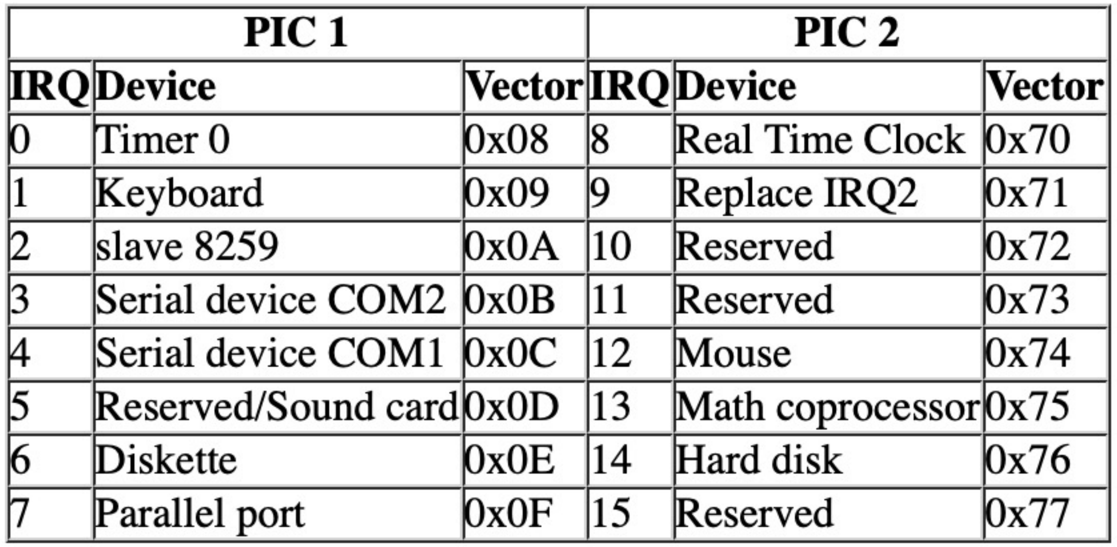
PC中断硬件

The 8259 PIC



PIC	Controller Register	Interrupt Mask Register
PIC1	0x20	0x21
PIC2	0xA0	0xA1

Table 4: PIC I/O ports.



PIC 1			PIC 2		
IRQ	**Device**	**Vector**	**IRQ**	**Device**	**Vector**
0	Timer 0	0x08	8	Real Time Clock	0x70
1	Keyboard	0x09	9	Replace IRQ2	0x71
2	slave 8259	0x0A	10	Reserved	0x72
3	Serial device COM2	0x0B	11	Reserved	0x73
4	Serial device COM1	0x0C	12	Mouse	0x74
5	Reserved/Sound card	0x0D	13	Math coprocessor	0x75
6	Diskette	0x0E	14	Hard disk	0x76
7	Parallel port	0x0F	15	Reserved	0x77

中断概念1：中断源（Source of Interrupts）

中断分类方式1：硬件中断 v.s. 软件中断

Hardware interrupt

- Caused by hardware.

Software Interrupt

- Invoked by the use of INT instruction.

中断概念1：中断源（Source of Interrupts）

中断分类方式2：外部中断 v.s. 内部中断

External Sources (Device interrupt)

- the mouse is moved
- A key is typed
- A network packet arrives
- The hard drive has completed a read/write operation
- A CD has been inserted into the CD drive.
- …

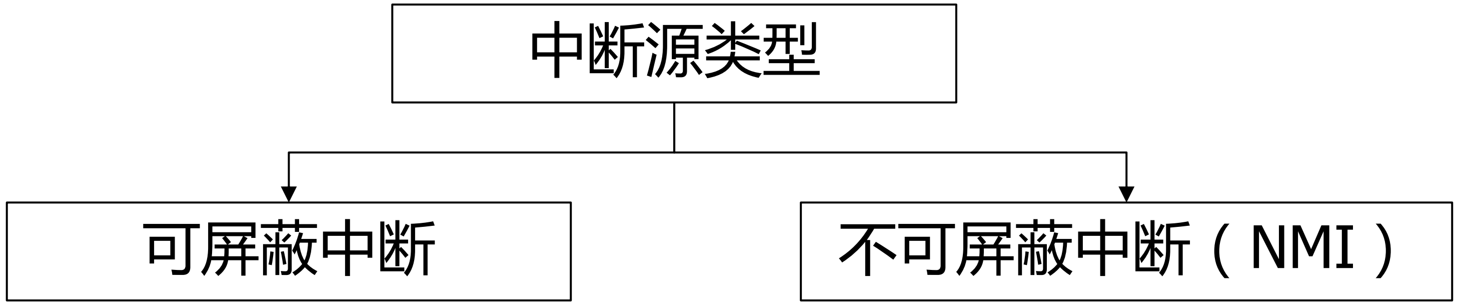
Internal Sources

- 地址非法
- 页故障
- 内部时钟中断
- 浮点数协处理器错误
- …

中断处理程序

中断概念1：中断源（Source of Interrupts）

中断分类方式3：可屏蔽中断 v.s. 不可屏蔽中断



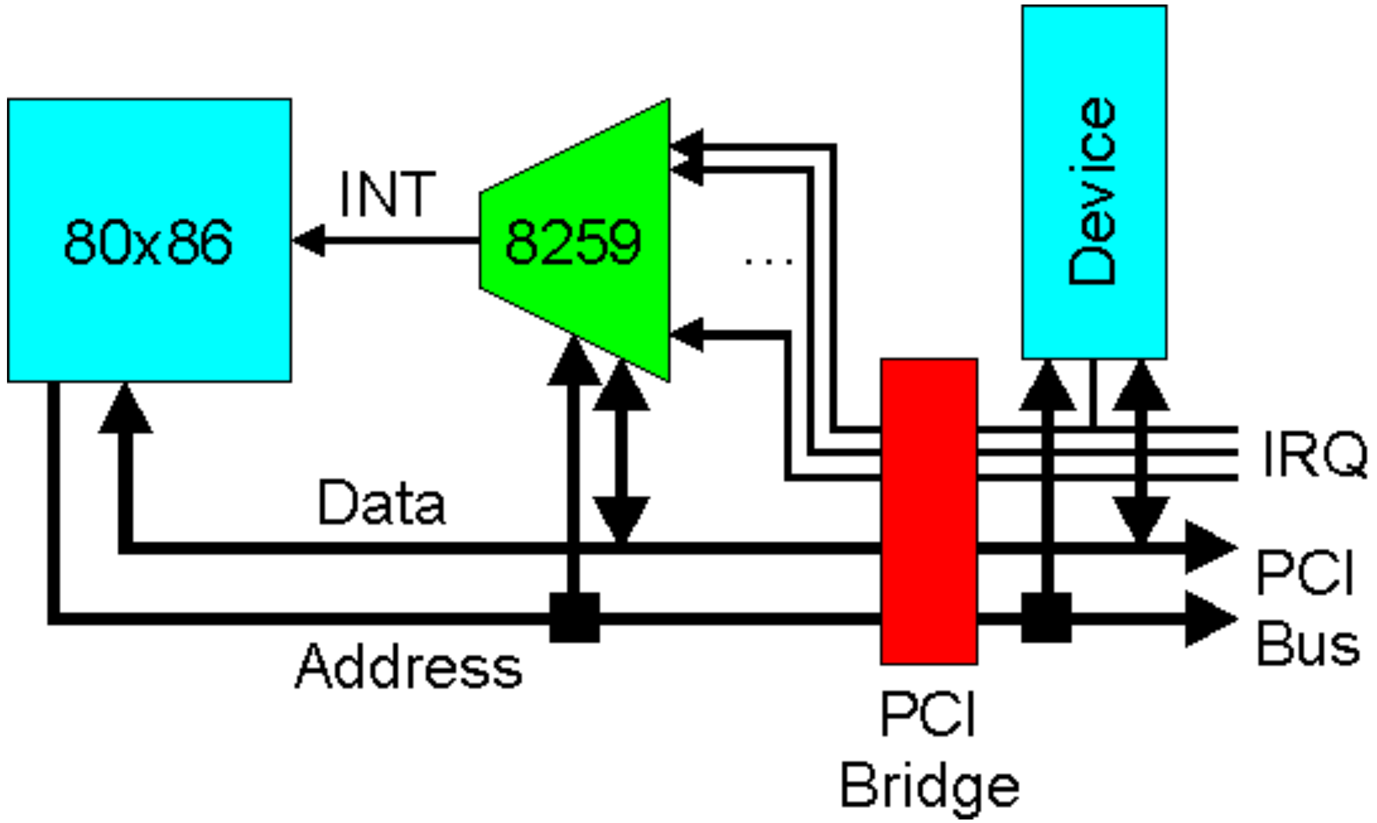
大部分的中断属于可屏蔽中断

典型的NMI：
- RAM parity error in a PC
- Division by Zero error

NMIs occur for RAM errors and unrecoverable hardware problems.

中断处理程序

中断概念2：中断控制器

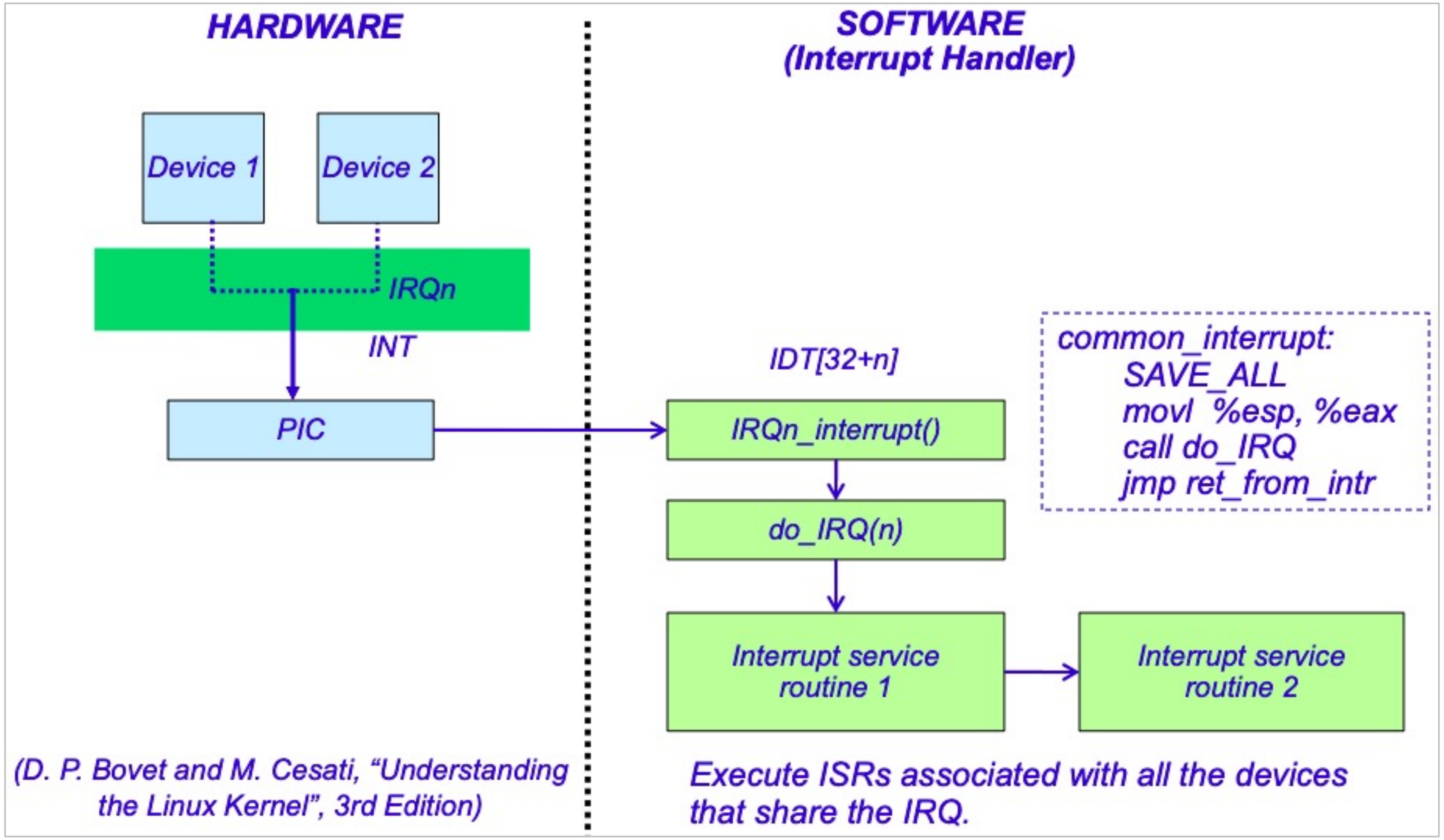


8259：Intel 80x86的中断控制器

中断处理程序

中断概念2：中断控制器

80x86/Linux环境下：中断控制器PIC与interrupt handler的协同

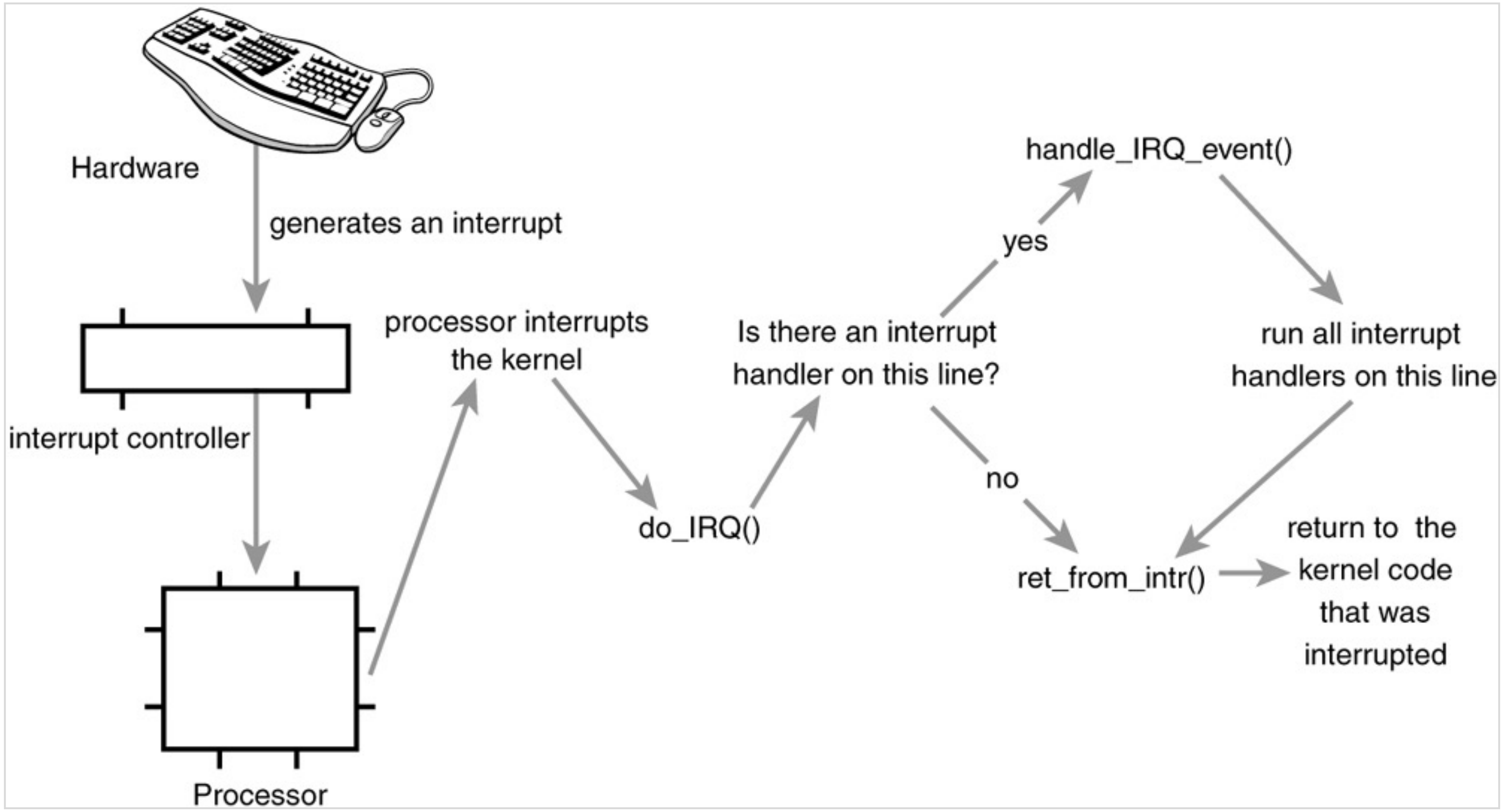


```
common_interrupt:
    SAVE_ALL
    movl  %esp, %eax
    call do_IRQ
    jmp ret_from_intr
```

Execute ISRs associated with all the devices that share the IRQ.

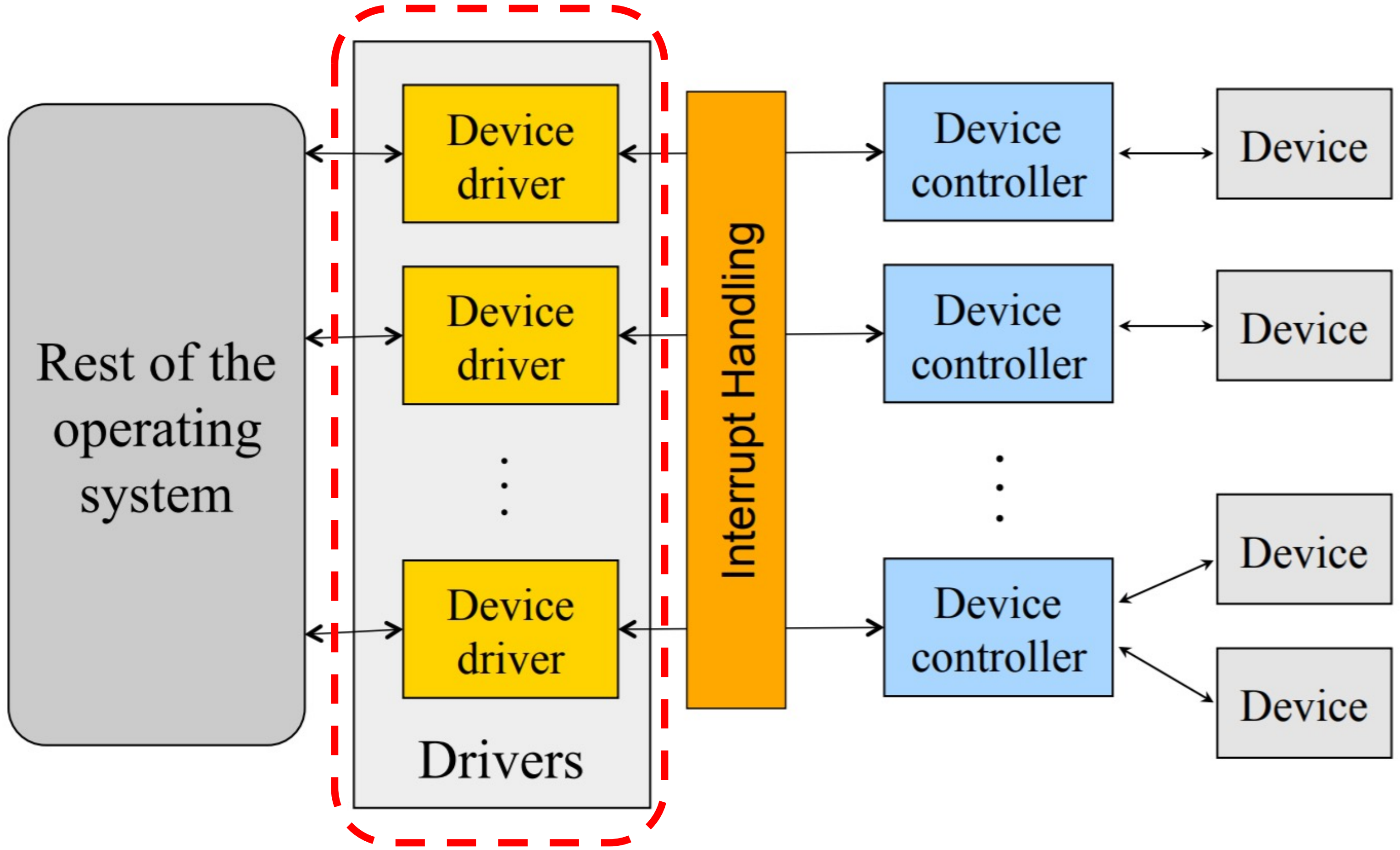
(D. P. Bovet and M. Cesati, “Understanding the Linux Kernel”, 3rd Edition)

Keyboard Interrupt Handling in Linux


Hardware
generates an interrupt
interrupt controller
Processor
processor interrupts the kernel
do_IRQ()
Is there an interrupt handler on this line?
yes
handle_IRQ_event()
run all interrupt handlers on this line
no
ret_from_intr()
return to the kernel code that was interrupted

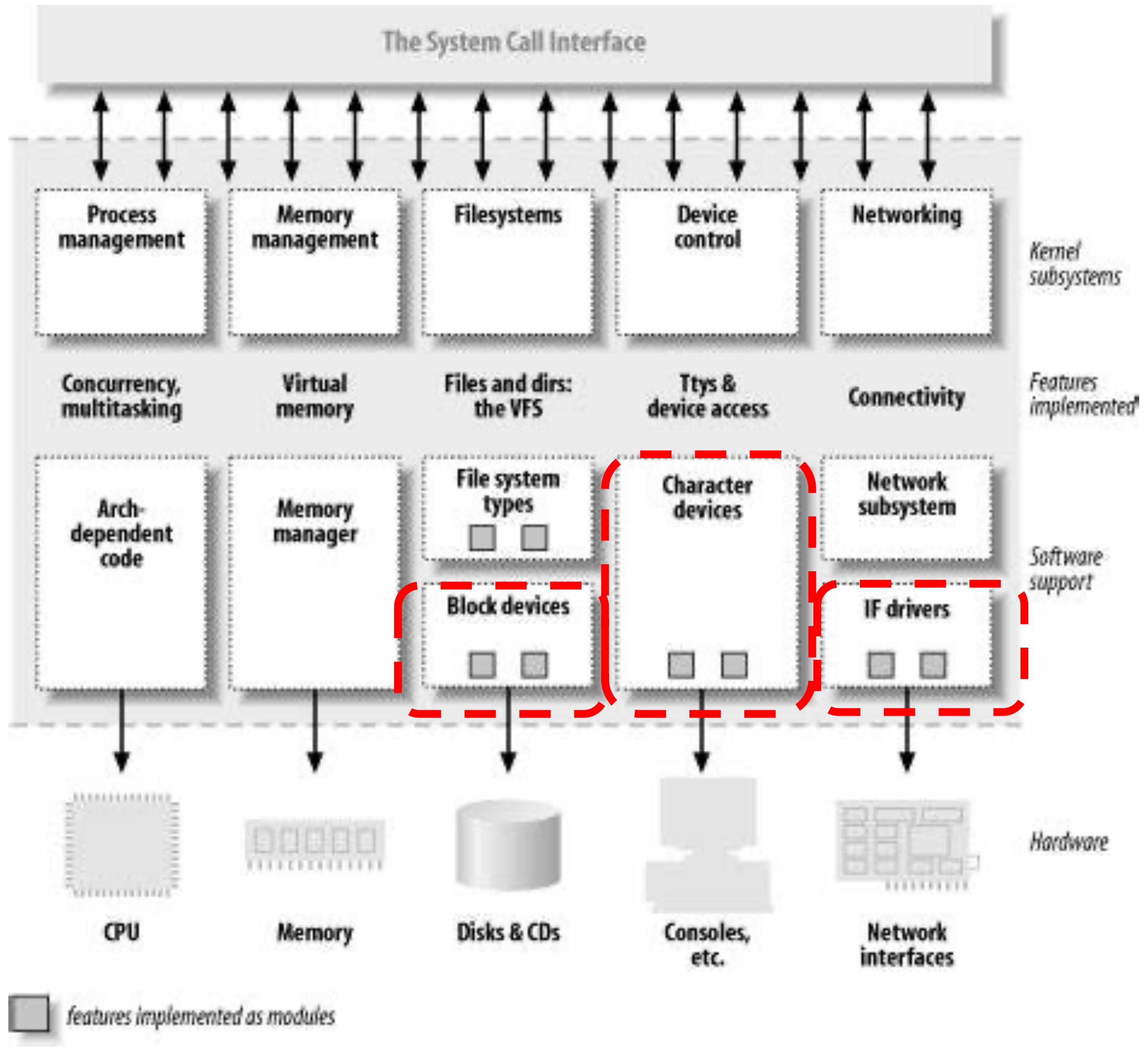
设备驱动程序

Device Drivers



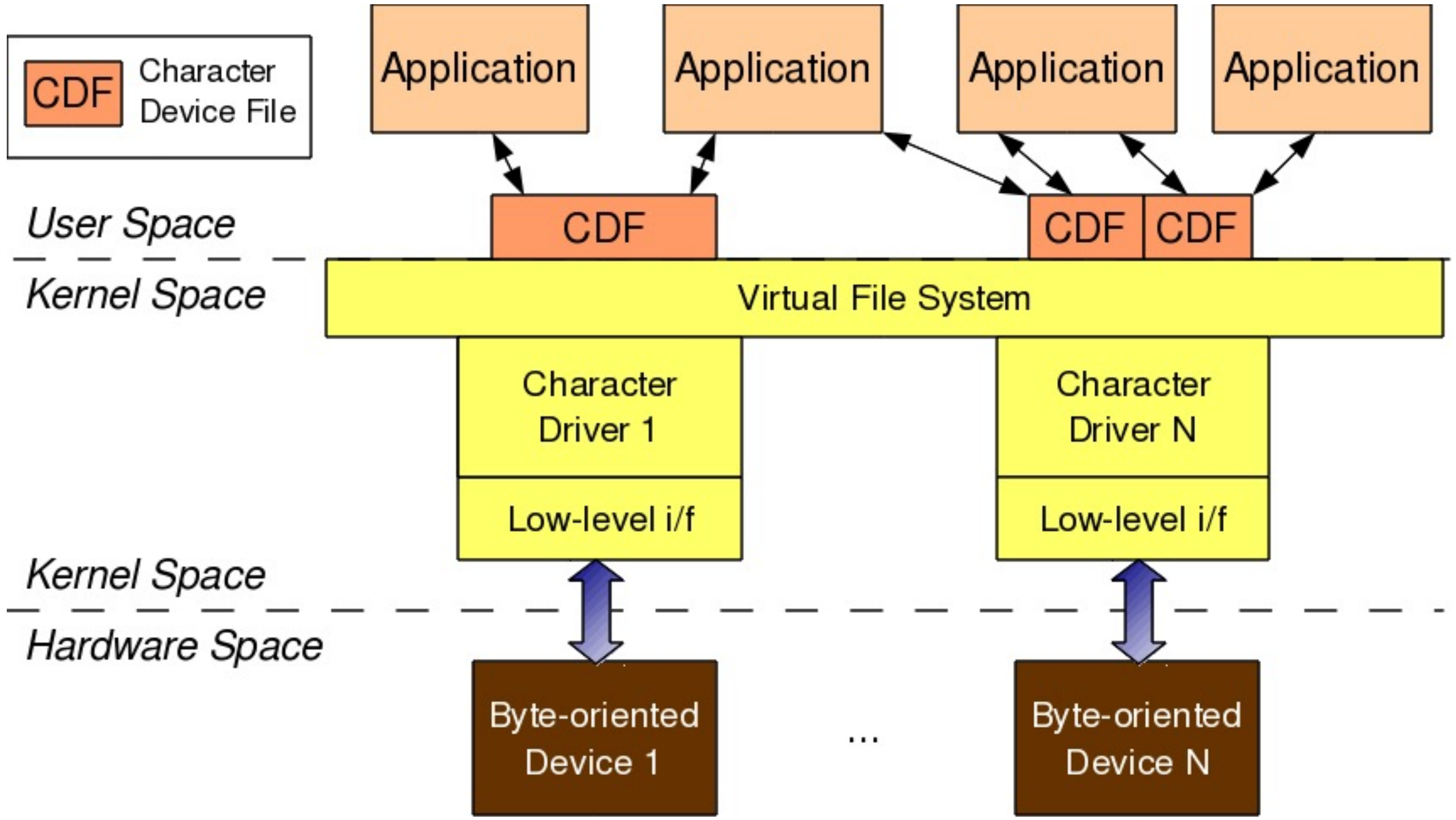
设备驱动程序

Linux Device Driver Modules



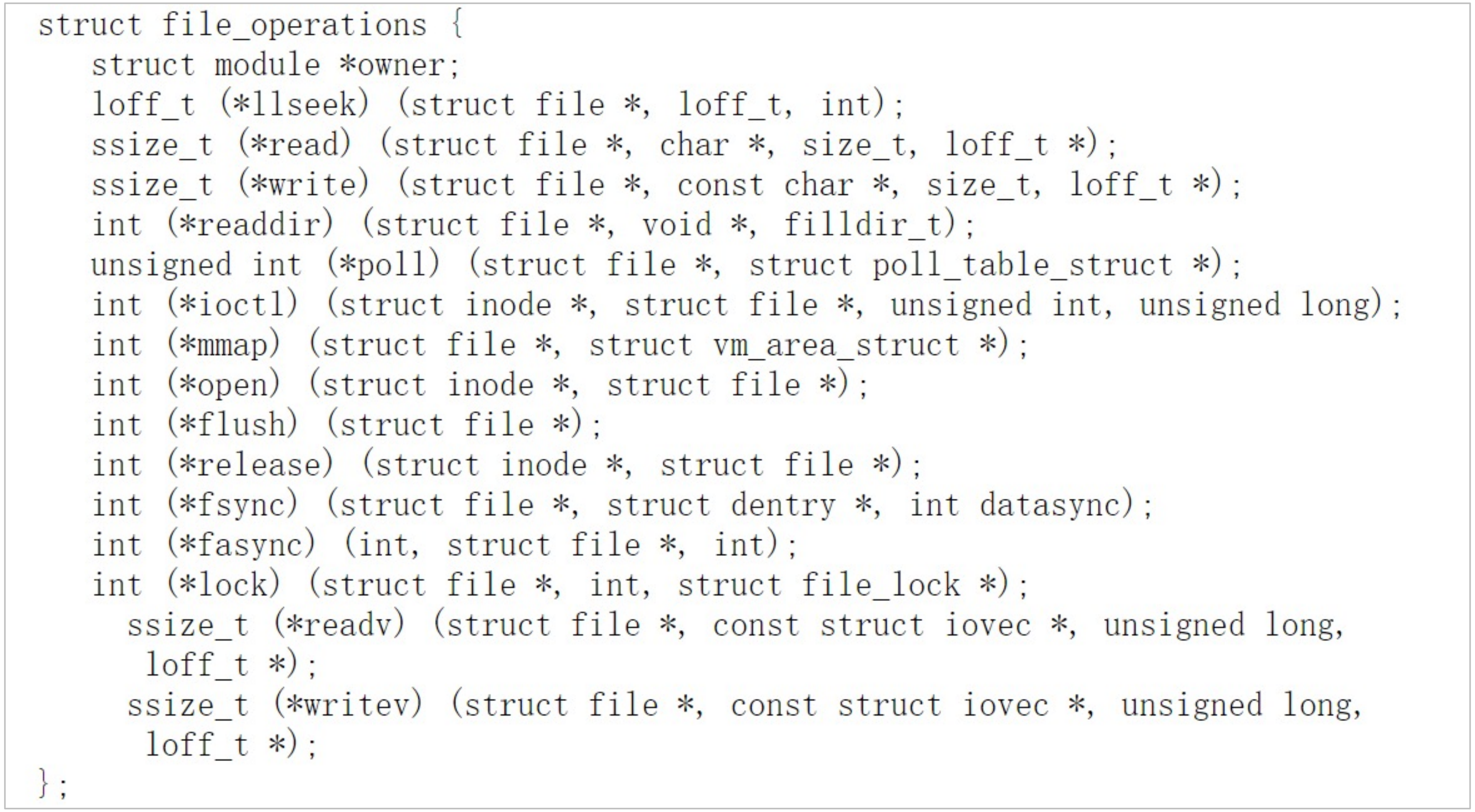
三大类驱动：
（1）字符设备驱动
（2）块设备驱动
（3）网络设备驱动

Linux Character Device



Linux Device File Operations

Linux Kernel 2.4.2



```
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
    int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *);
    int (*release) (struct inode *, struct file *);
    int (*fsync) (struct file *, struct dentry *, int datasync);
    int (*fasync) (int, struct file *, int);
    int (*lock) (struct file *, int, struct file_lock *);
      ssize_t (*readv) (struct file *, const struct iovec *, unsigned long,
       loff_t *);
      ssize_t (*writev) (struct file *, const struct iovec *, unsigned long,
       loff_t *);
};
```

Device Drivers Behavior

Check input parameters for validity, and translate them to device specific language

Check if device is free (wait or block if not)

Issue commands to control device

- Write them into device controller’ s registers
- Check after each command if device is ready for next (wait or block if not)

Block or wait for controller to finish work

Check for errors, and pass data to device-independent software

Return status information

Process next queued request, or block waiting for next

Challenges:
-Must be reentrant (can be called by an interrupt while running)
-Handle hot-pluggable devices and device removal while running
-Complex, and many of them; bugs in them can crash system

Device Drivers Behavior

设备驱动程序应具备的功能：

1）接收由设备独立性软件层发来的命令和参数，并将命令中的抽象要求转换为具体要求

例如：将磁盘块号转换为磁盘的盘面、磁道号及扇区号。

设备驱动程序

Device Drivers Behavior

设备驱动程序应具备的功能：

1）接收由设备独立性软件层发来的命令和参数，并将命令中的抽象要求转换为具体要求

2）检查用户I/O请求的合法性，了解I/O设备的状态，传递有关参数，设置设备的工作方式

设备驱动程序

Device Drivers Behavior

设备驱动程序应具备的功能：

1）接收由设备独立性软件层发来的命令和参数，并将命令中的抽象要求转换为具体要求

2）检查用户I/O请求的合法性，了解I/O设备的状态，传递有关参数，设置设备的工作方式

3）发出I/O命令

如果设备空闲，便立即启动I/O设备去完成指定的I/O操作;
如果设备处于忙碌状态，则将请求者的请求块挂在设备队列上等待

设备驱动程序

Device Drivers Behavior

设备驱动程序应具备的功能：

1）接收由设备独立性软件层发来的命令和参数，并将命令中的抽象要求转换为具体要求

2）检查用户I/O请求的合法性，了解I/O设备的状态，传递有关参数，设置设备的工作方式

3）发出I/O命令

4）及时响应由控制器或通道发来的中断请求，并根据其中断类型调用相应的中断处理程序进行处理

Device-Independence Software & Uniform Naming

在**应用程序**中，使用**逻辑设备名称**来请求使用某类设备;
将对物理设备访问的细节封装在I/O相关系统调用的实现内

The name of a file or device should be a string or integer and not depend on the device in any way

Example (Linux Device Names):

- The first floppy drive is named /dev/fd0.
- The second floppy drive is named /dev/fd1.
- The first SCSI disk (SCSI ID address-wise) is named /dev/sda.
- The second SCSI disk (address-wise) is named /dev/sdb, and so on.
- The first SCSI CD-ROM is named /dev/scd0, also known as /dev/sr0.
- The master disk on IDE primary controller is named /dev/hda.
- The slave disk on IDE primary controller is named /dev/hdb.

1-I/O软件 设备无关软件

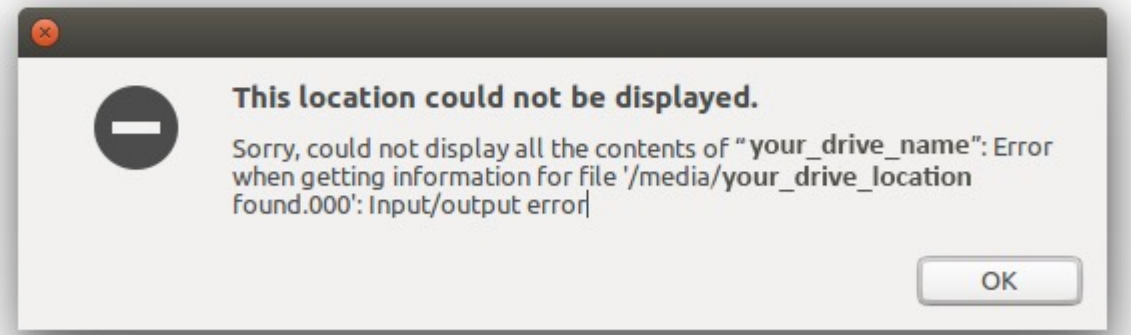
Error Handling

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

An OS that uses protected memory can guard against many kinds of hardware and application errors so that a complete system failure is not the usual result of each minor mechanical glitch.

Devices, and I/O transfers can fail in many ways, either :

- for transient reasons, as when a network becomes overloaded
- or for permanent reasons, as when a disk controller becomes defective.


This location could not be displayed.

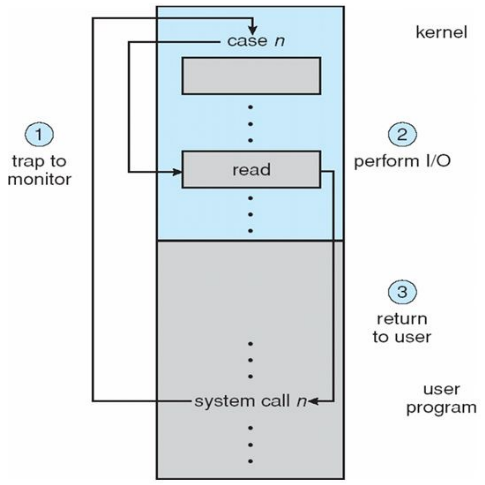
Sorry, could not display all the contents of "your_drive_name": Error when getting information for file '/media/your_drive_location found.000': Input/output error

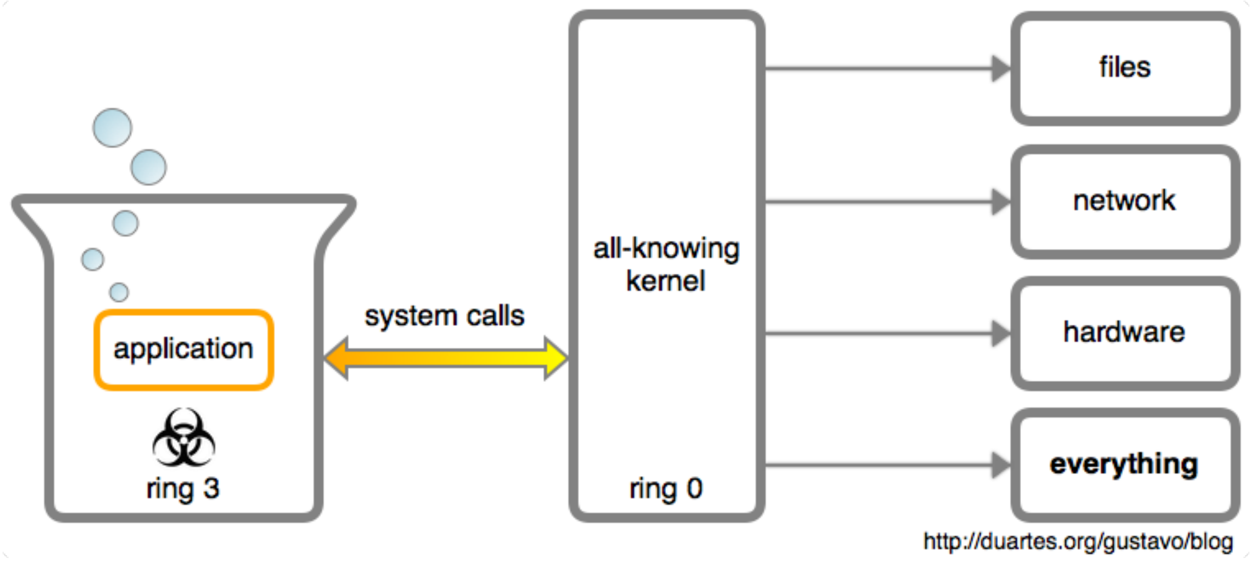
I/O Protection

- All I/O instructions are privileged instructions
- All I/O is performed through system calls.

Errors and the issue of protection are closely related. A user process may attempt to issue illegal I/O instructions to disrupt the normal function of a system. We can use the various mechanisms to ensure that such disruption cannot take place in the system. To prevent illegal I/O access, we define all I/O instructions to be privileged instructions. The user cannot issue I/O instruction directly.

Use of a System Call to Perform I/O





SPOOLING & Device Reservation

SPOOLING

a technique in which an intermediate device like a disk, is interposed between a process and a low speed or buffer limited I/O device.

独占设备，被改造为共享设备

借助于磁盘缓冲区，营造出多个虚拟设备的状态
=> SPOOLING被称为“虚拟设备技术”

Device Reservation

provides exclusive device access.
Allocation of devices when required by the processes and that device when it is no longer needed.
It prevents deadlock.
Many operating systems provide features that enable processes to coordinate and give exclusive access to them.

提供对设备的独占使用方式

预防I/O操作的死锁

I/O System Calls

read

write

recv/recvfrom/recvmsg

poll

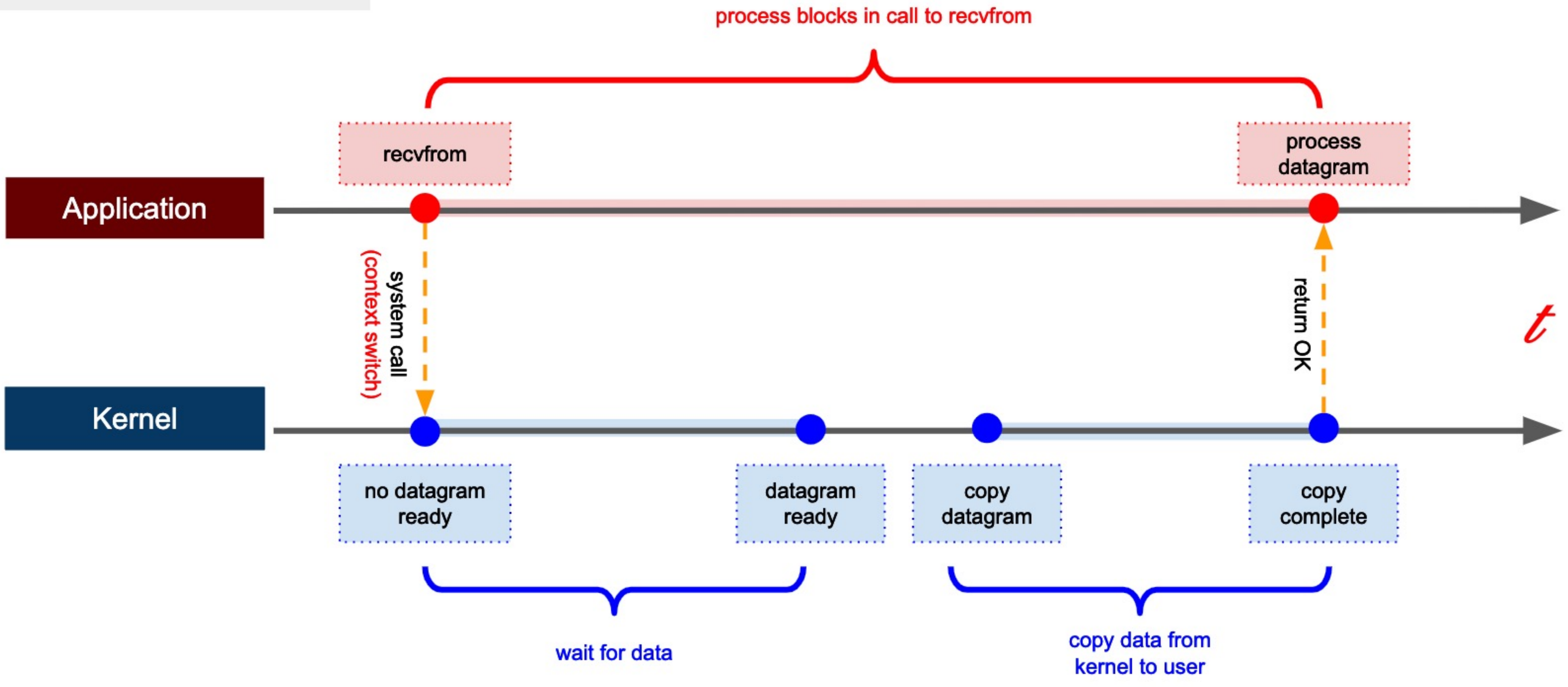
epoll

I/O系统调用

I/O模型

1-Blocking I/O (BIO)

Fig 6.1 Blocking I/O Model

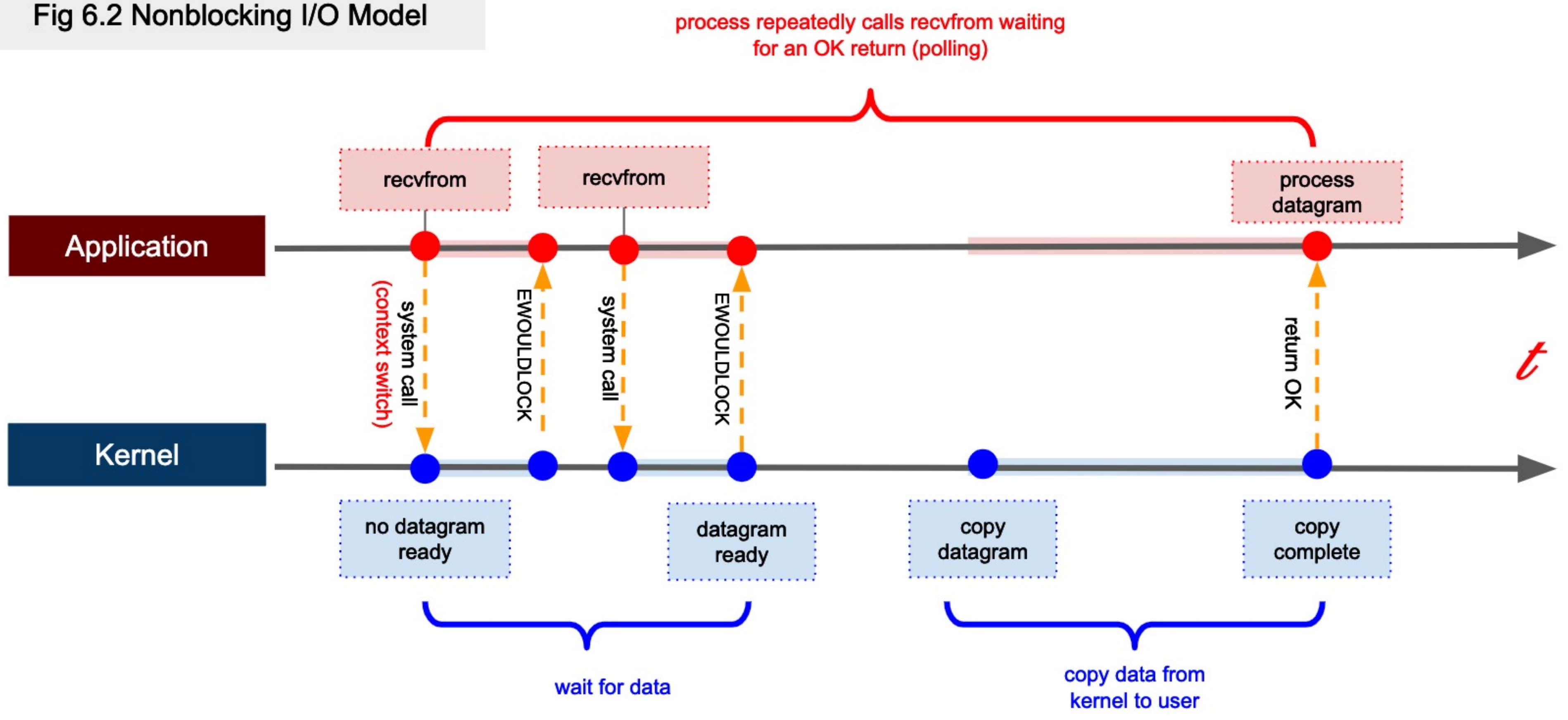


I/O系统调用

I/O模型

2-Non-Blocking I/O

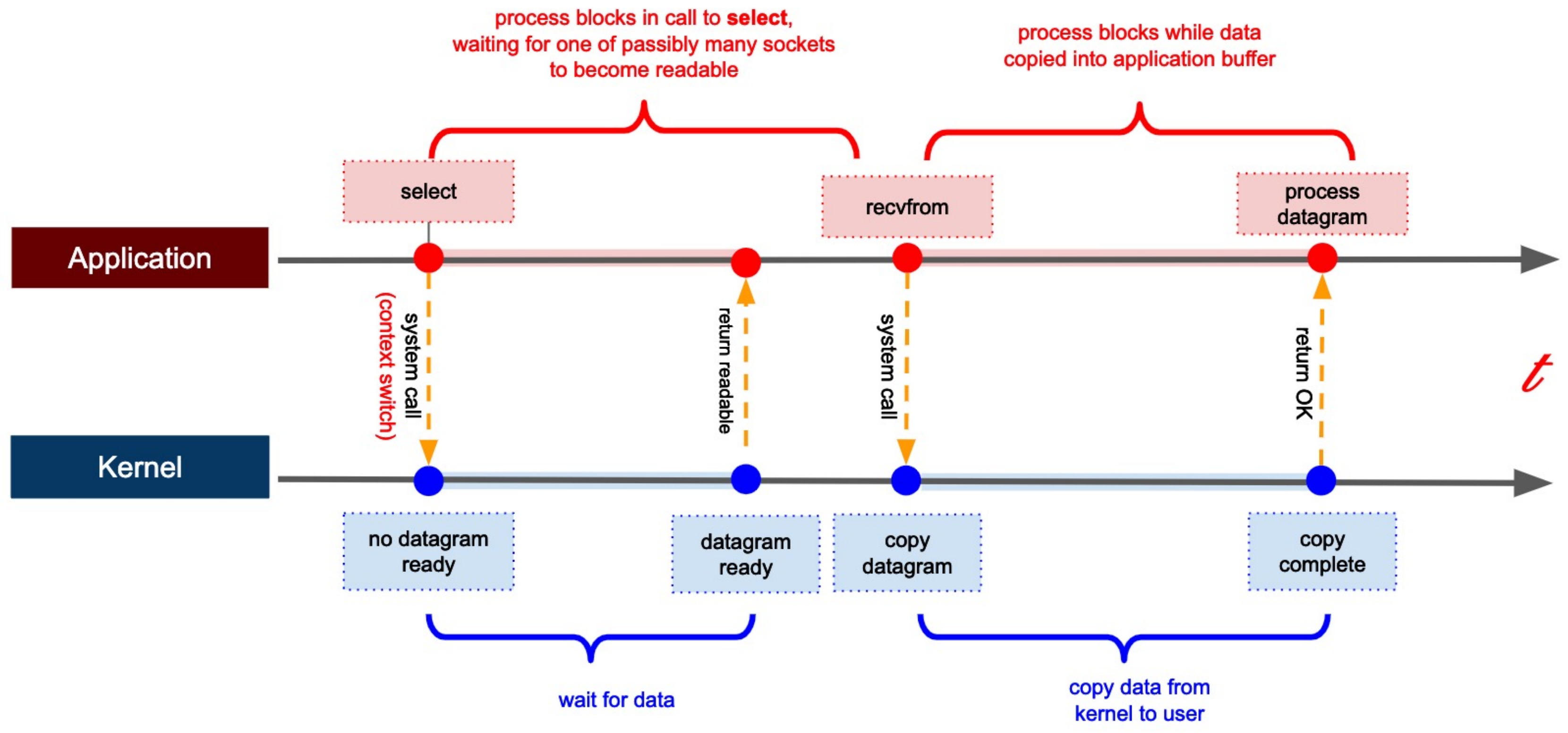
Fig 6.2 Nonblocking I/O Model



I/O模型

3-I/O Multiplexing

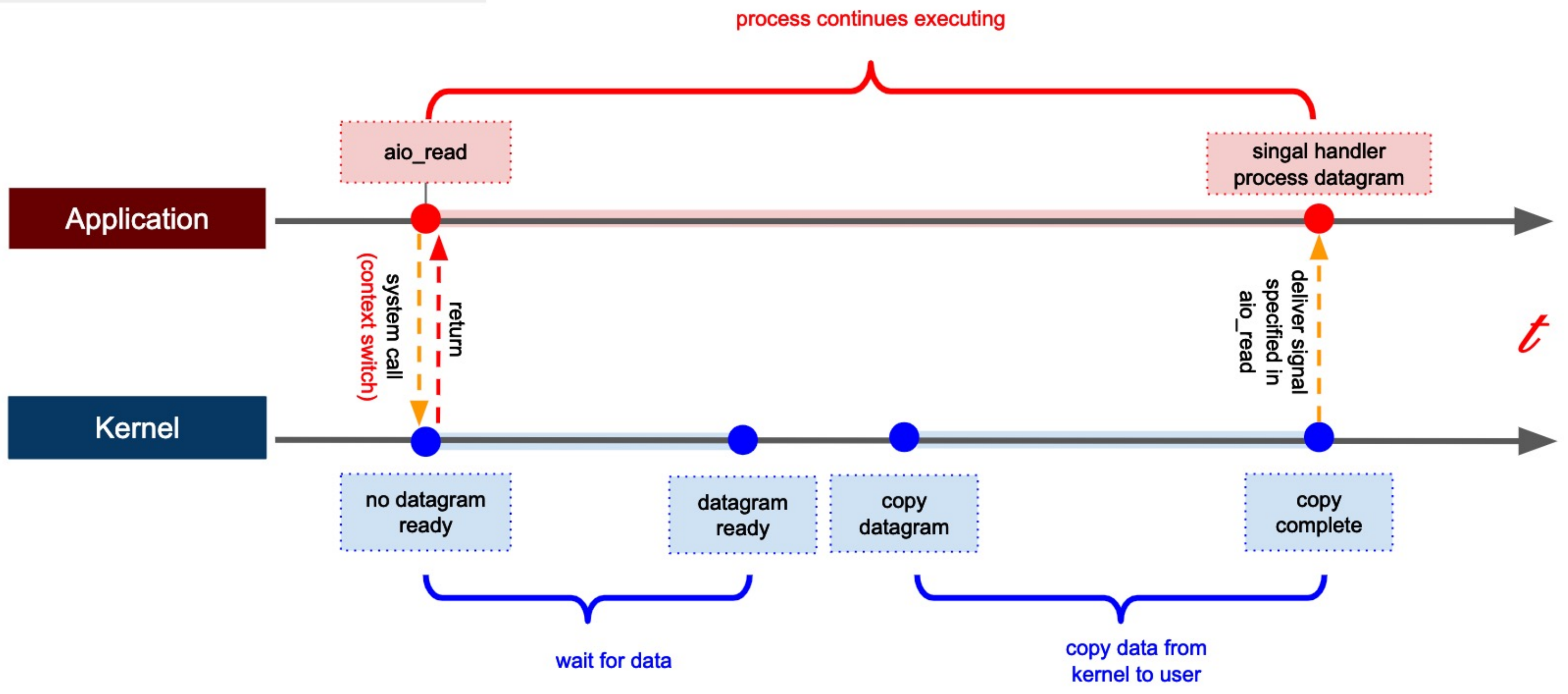
Fig 6.3 I/O Multiplexing Model


process blocks in call to **select**, waiting for one of passibly many sockets to become readable
process blocks while data copied into application buffer
select
recvfrom
process datagram
Application
Kernel
system call
(context switch)
return readable
system call
return OK
t
no datagram ready
datagram ready
copy datagram
copy complete
wait for data
copy data from kernel to user

I/O模型

4-Asynchronous I/O (AIO)

Fig 6.5 Asynchronous I/O model



I/O模型

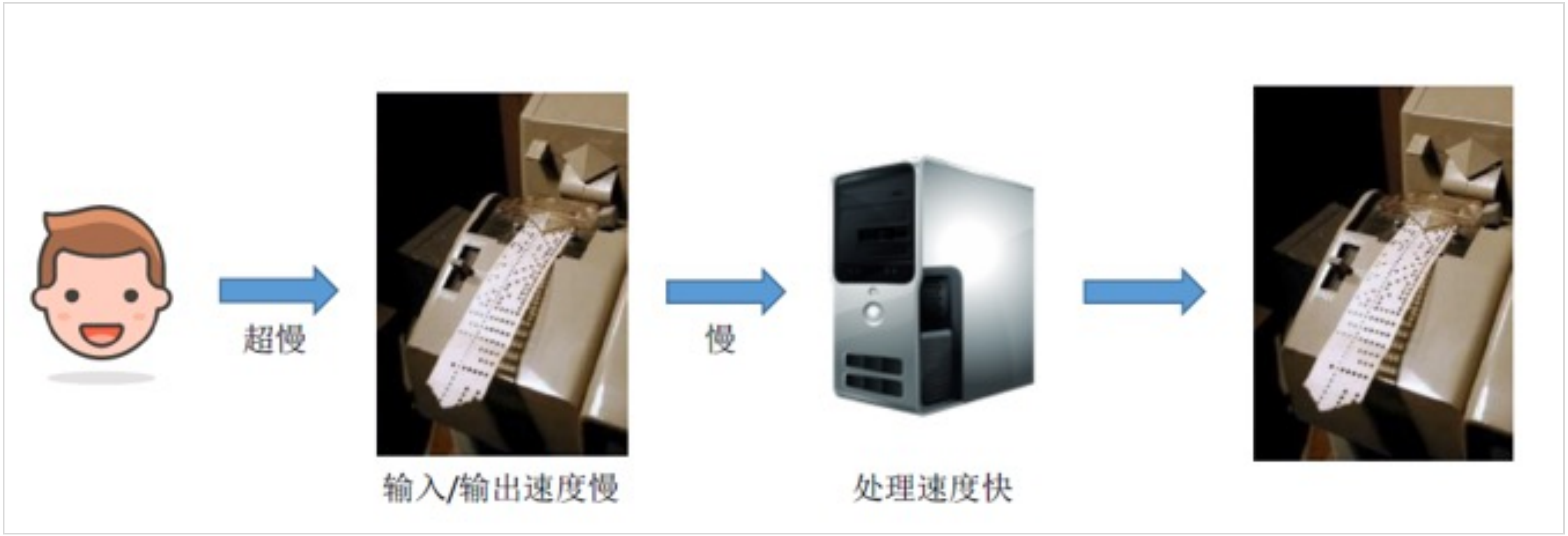
	Blocking	Non-Blocking
Synchronous	Read / Write	Read / Write (O_NONBLOCK)
Asynchronous	I/O Multiplexing (select / poll)	AIO

SPOOLING技术 02

SPOOLING

相关背景

没有出现操作系统之前，I/O依赖手工操作

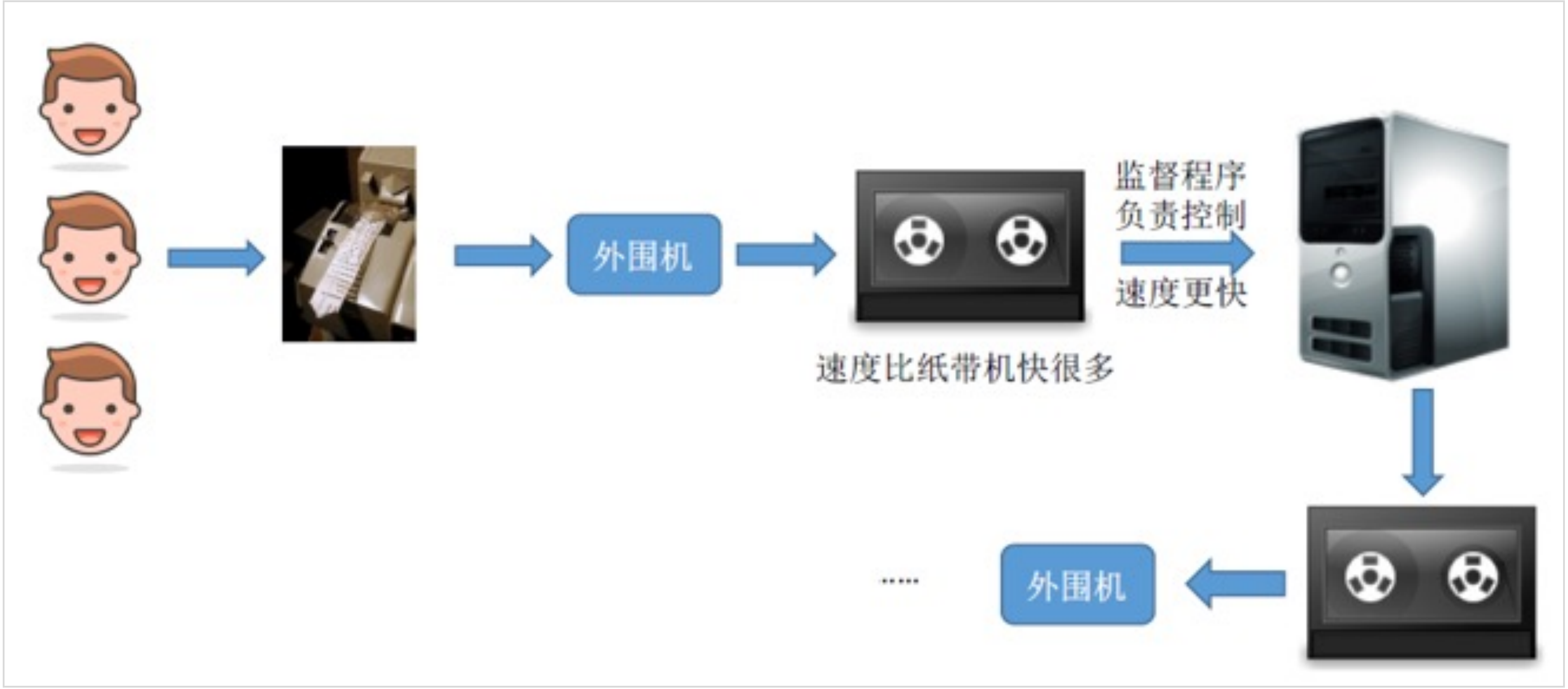


效率低：IO速度非常慢，主机浪费很多时间等待I/O

相关背景

脱机技术

通过外围机将数据预先传送到较快速的磁带，再由主机上专门的监督程序从磁带传入主机磁盘



目的：缓解IO设备与CPU速度不匹配的矛盾

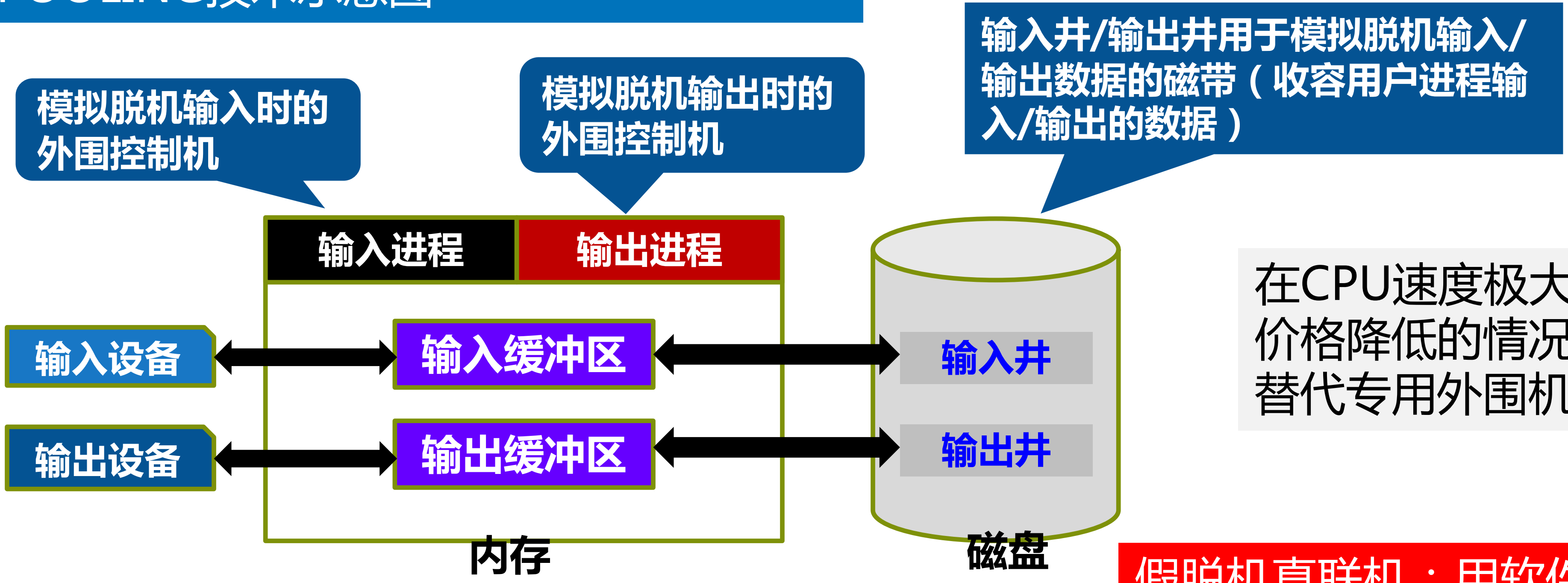
SPOOLING概念

SPOOLING（外部设备联机并行操作）

-Simultaneous Peripheral Operations On-line

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

SPOOLING技术示意图


模拟脱机输入时的外围控制机
模拟脱机输出时的外围控制机
输入井/输出井用于模拟脱机输入/输出数据的磁带（收容用户进程输入/输出的数据）
输入进程
输出进程
输入设备
输出设备
输入缓冲区
输出缓冲区
输入井
输出井
内存
磁盘

在CPU速度极大提升、磁盘普及价格降低的情况下，以软件模拟替代专用外围机作用

假脱机真联机：用软件的方式模拟脱机技术

SPOOLING技术的特点

SPOOLING技术的主要特性

具有缓解CPU与I/O速度严重不匹配的矛盾的重要意义

通过设立输入井和输出井作为缓冲，从对低速I/O设备进行的I/O操作变为对输入井或输出井的操作，使得CPU与I/O设备可以异步并发模式工作，解放了CPU

具有将独占设备改造为共享设备的效果

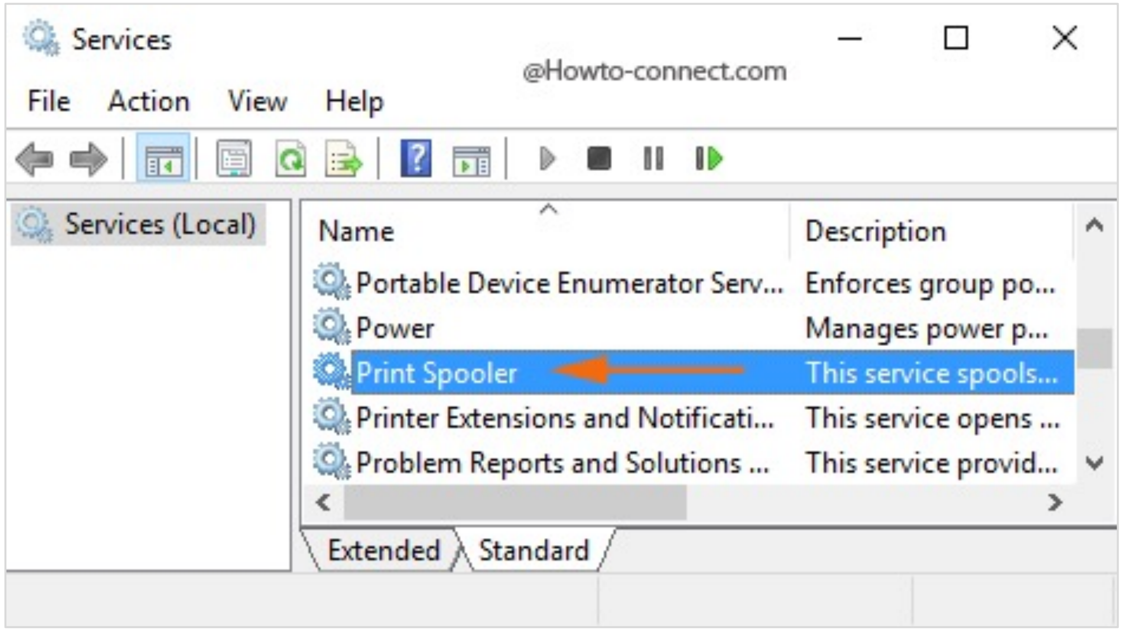
实现了虚拟设备的效果（改造成的共享设备=虚拟设备）

多个进程同时使用一独享设备，而对每一进程而言，都认为自己独占这一设备，从而实现了设备的虚拟分配

SPOOLING示例

示例: 将打印机模拟成共享设备

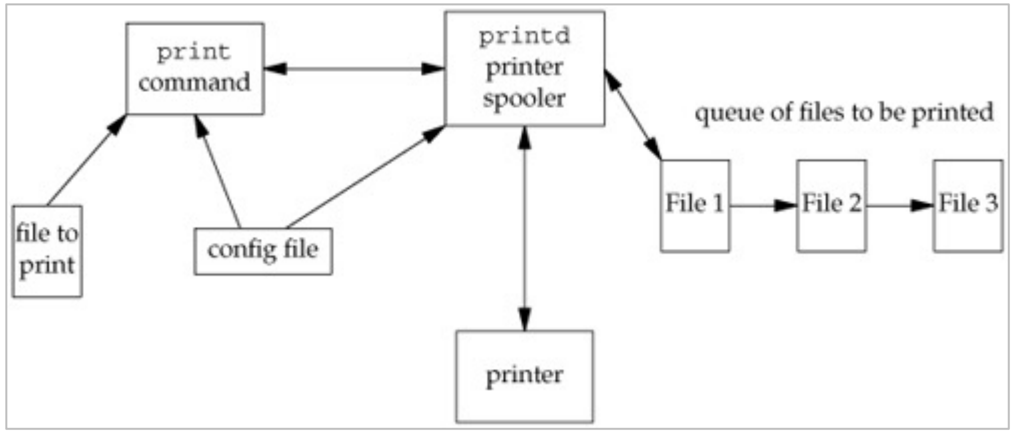
打印机是典型的独占型设备
现代OS会通过SPOOLING技术将打印机模拟成共享设备。


Services
@Howto-connect.com
File Action View Help
Services (Local)

Name	Description
Portable Device Enumerator Serv...	Enforces group po...
Power	Manages power p...
Print Spooler	This service spools...
Printer Extensions and Notificati...	This service opens ...
Problem Reports and Solutions ...	This service provid...

Extended Standard

Win10 spooler service setup

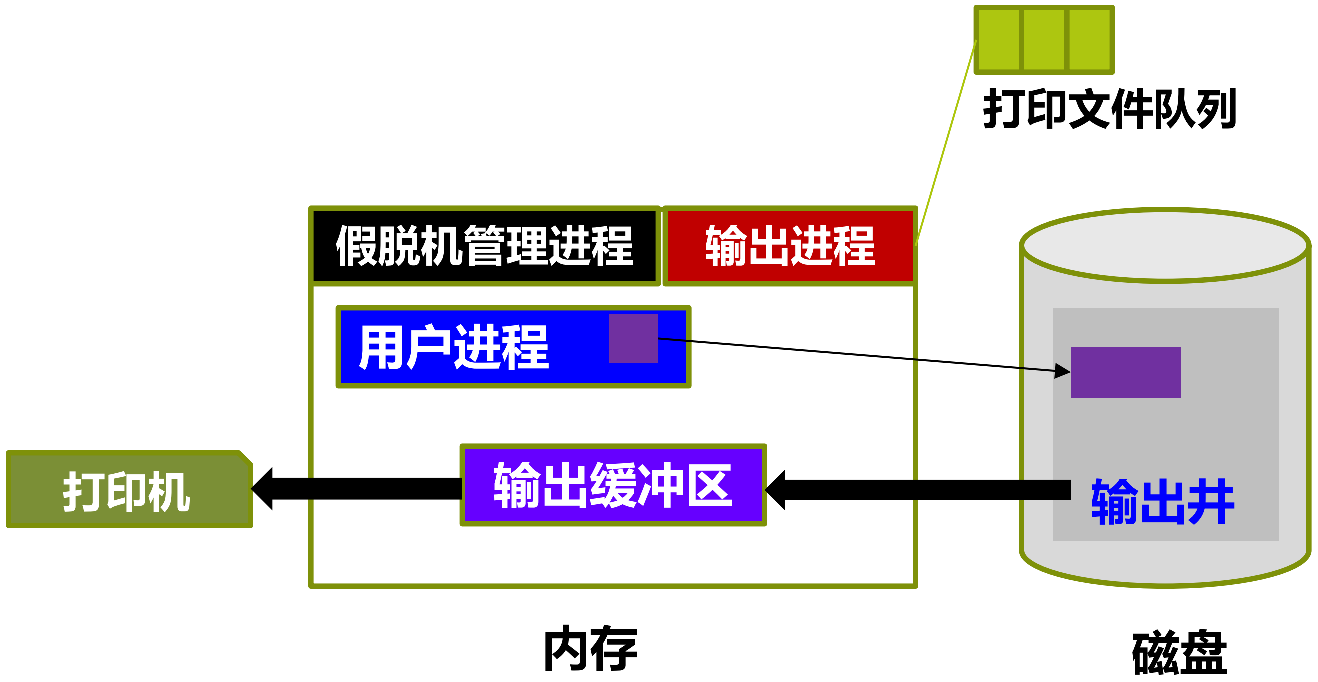


Unix Printer Spooler workflow

SPOOLING示例

示例: 将打印机模拟成共享设备

打印机是典型的独占型设备
现代OS会通过SPOOLING技术将打印机模拟成共享设备。



Summary

小结：

- I/O软件
- SPOOLING技术

设备无关性

设备无关软件层

呈现出的是设备无关性

具备设备无关性的软件层能够做到：
应用在访问物理设备时无需知道设备的物理名称和具体的配置信息，而仅需给出设备名称，就可以访问设备并其发出I/O请求

实现设备无关性，需要I/O软件通过设备无关软件层，实现从逻辑设备名到物理设备的管理数据结构的映射关系；

中断处理在I/O软件中的重要性

谢 谢！
Thank you!