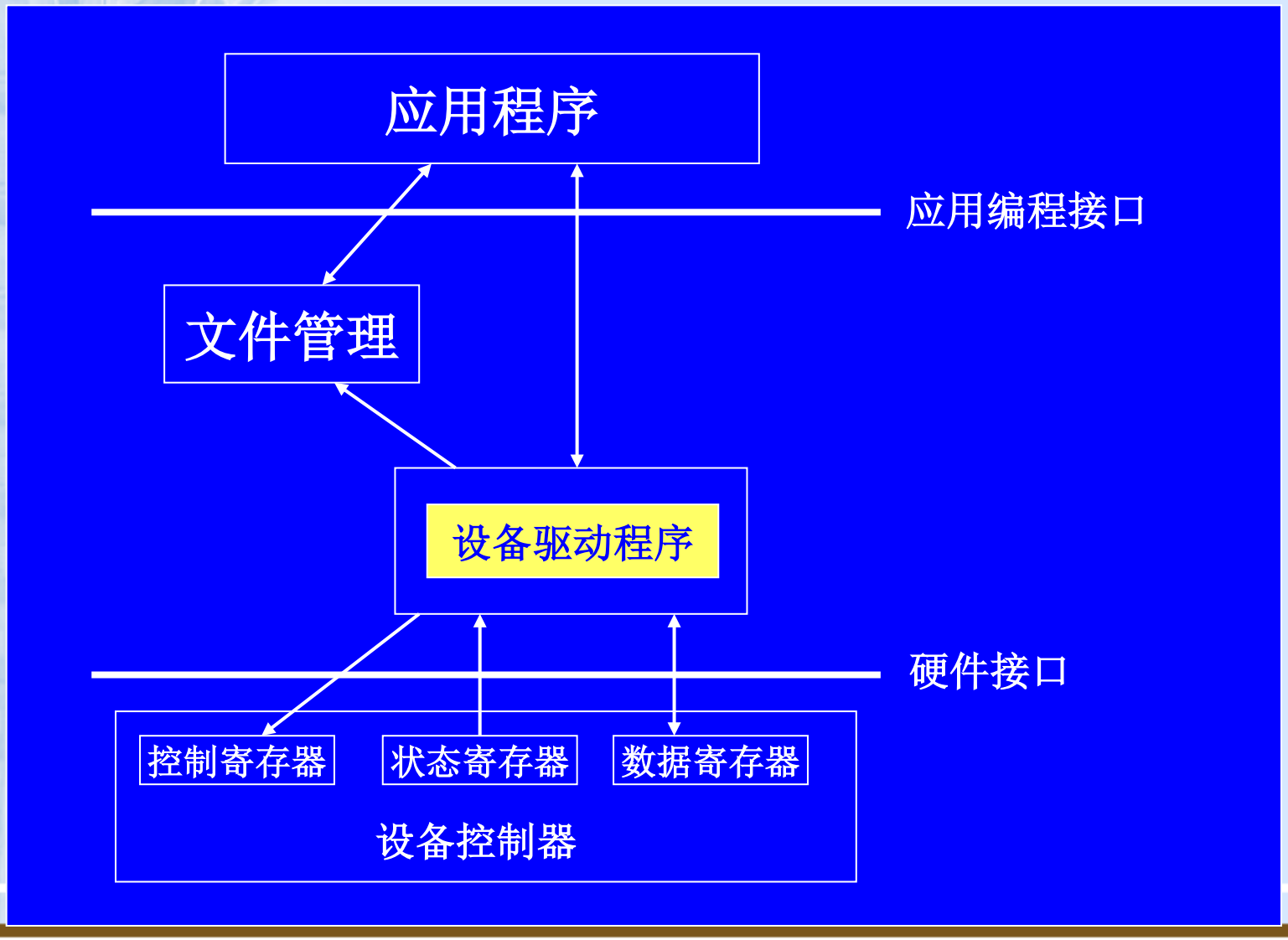
火龙果•整理
uml.org.cn

第8讲 I/O系统

- 概述
- I/O硬件特点
- I/O软件的组成
- 相关技术
- 设备处理

火龙果•整理
uml.org.cn

I/O管理示意



火龙果•整理
uml.org.cn

一、概述

1. I/O的特点

CPU与I/O的速度差别大

→尽量使两者交叠运行

→减少由于速度差异造成的整体性能开销

（1）I/O性能经常成为系统性能的瓶颈

（2）操作系统庞大复杂的原因之一：

资源多、杂，并发，均来自I/O

- 外设种类繁多，结构各异
- 输入输出数据信号类型不同
- 速度差异很大

（3）与其他功能联系密切，特别是文件系统

火龙果•整理
uml.org.cn

2. 设备的分类(1/2)

（1）按数据组织分

- 块设备

  以数据块为单位存储、传输信息

  传输速率较高、可寻址（随机读写）

- 字符设备

  以字符为单位存储、传输信息

  传输速率低、不可寻址

火龙果•整理
uml.org.cn

设备的分类(2/2)

（2）按资源分配角度分

- 独占设备

  在一段时间内只能有一个进程使用的设备，一般为低速I/O设备（如打印机，磁带等）

- 共享设备

  在一段时间内可有多个进程共同使用的设备，多个进程以交叉的方式来使用设备，其资源利用率高（如硬盘）

- 虚设备

  在一类设备上模拟另一类设备，常用共享设备模拟独占设备，用高速设备模拟低速设备，被模拟的设备称为虚设备

目的：将慢速的独占设备改造成多个用户可共享的设备，提高设备的利用率

（实例：SPOOLing技术，利用虚设备技术——用硬盘模拟输入输出设备）

火龙果•整理
uml.org.cn

3. 设备管理的目标和任务(1/3)

（1）按照用户的请求，控制设备的各种操作，完成I/O设备与内存之间的数据交换，最终完成用户的I/O请求

- 设备分配与回收

  记录设备的状态

  根据用户的请求和设备的类型，采用一定的分配算法，选择一条数据通路
- 执行设备驱动程序，实现真正的I/O操作
- 设备中断处理：处理外部设备的中断
- 缓冲区管理：管理I/O缓冲区

火龙果•整理
uml.org.cn

设备管理的目标和任务(2/3)

（2）建立方便、统一的独立于设备的接口

方便性：向用户提供使用外部设备的方便接口，使用户编程时不考虑设备的复杂物理特性

统一性：对不同的设备采取统一的操作方式，在用户程序中使用的是逻辑设备

逻辑设备与物理设备、屏蔽硬件细节（设备的物理细节，错误处理，不同I/O的差异性）

火龙果•整理
uml.org.cn

设备管理的目标和任务(3/3)

（3）充分利用各种技术（通道，中断，缓冲，异步I/O等）提高CPU与设备、设备与设备之间的并行工作能力，充分利用资源，提高资源利用率

- 并行性
- 均衡性（使设备充分忙碌）

（4）保护

设备传送或管理的数据应该是安全的、不被破坏的、保密的

火龙果•整理
uml.org.cn

二、I/O硬件特点

1. 设备组成(1/2)

I／O设备一般由机械和电子两部分组成

把这两部分分开处理，以提供更加模块化，更加通用的设计

（1）物理设备

机械部分是设备本身（物理装置）

（2）设备控制器

- 电子部分叫做设备控制器或适配器
- 完成设备与主机间的连接和通信
- 在小型和微型机中，它常采用印刷电路卡插入计算机主板上的总线插槽
- 通过若干接口寄存器或接口缓冲区与CPU通信

火龙果•整理
uml.org.cn

设备组成(2/2)

电子部分完成的工作

- （端口）地址译码
- 按照主机与设备之间约定的格式和过程
  - 接受计算机发来的数据和控制信号
  - 向主机发送数据和状态信号
- 将计算机的数字信号转换成机械部分能识别的模拟信号，或反之
- 实现设备内部硬件缓冲、数据加工等提高性能或增强功能

火龙果•整理
uml.org.cn

2.设备接口(1/2)

操作系统将命令写入控制器的接口寄存器（或接口缓冲区）中，以实现输入／输出，并从接口寄存器读取状态信息或结果信息

例如：IBM PC的软盘控制器可接收15条命令，READ、WRITE、FORMAT、SEEK、RECALIBRATE，命令可以带参数，它们被一起送入控制器的寄存器中

当控制器接受一条命令后，可独立于CPU完成指定操作，CPU可以转去执行其他运算；命令完成时，控制器产生一个中断，CPU响应中断，控制转给操作系统；CPU通过读控制器寄存器中的信息，获得操作结果和设备状态

火龙果•整理
uml.org.cn

设备接口(2/2)

控制器与设备之间的接口常常是一个低级接口。例如磁盘，可以按每个磁道8扇区，每个扇区512个字节进行格式化。实际从驱动器出来的却是一连串的位流，以一个头标（preamble）开始，然后是一个磁道的4096位（512×8），最后是检查和或错误校验码（Error—C—C：ECC）。头标是在对磁盘格式化时写上的，它包括柱面和扇区数，扇区的大小和类似的一些数据

控制器的任务是把串行的位流转换为字节块，并进行必要的错误修正：首先，控制器按位进行组装，然后存入控制器内部的缓冲区中形成以字节为单位的块。在对块验证检查和并证明无错误时，再将它复制到内存中

火龙果•整理
uml.org.cn

3. I/O端口地址(1/2)

- 接口电路中：多个寄存器
- 每个寄存器有唯一的一个地址，该地址称为I/O端口地址，是一个整数
- 所有I/O端口地址形成I/O端口空间(受到保护)

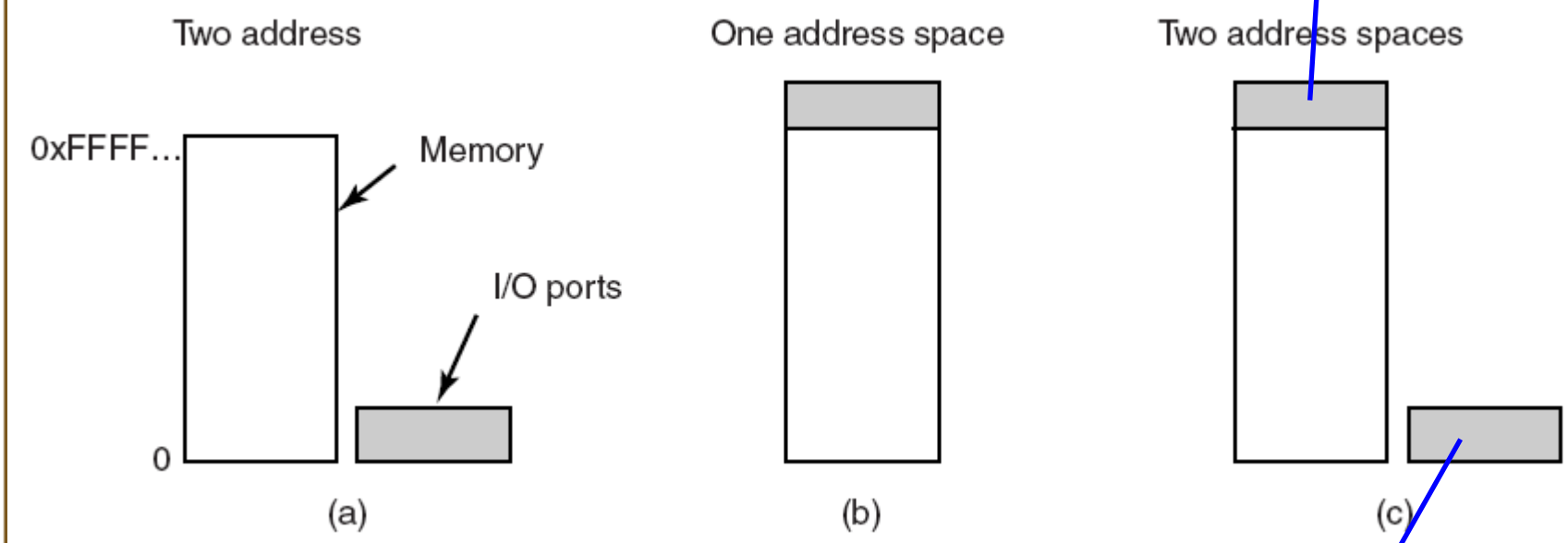
I/O指令形式与I/O地址是相互关联的

主要有两种形式：

- 内存映像编址（内存映像I/O模式）
- I/O独立编址（I/O专用指令）

火龙果•整理
uml.org.cn

I/O端口地址(2/2)



内存映射I/O的数据缓冲区

独立I/O端口的控制寄存器

火龙果•整理
uml.org.cn

I/O独立编址

- 分配给系统中所有端口的地址空间是完全独立的，与内存的地址空间没有关系
- 使用专门的I/O指令对端口进行操作
- 优点
  - 外部设备不占用内存的地址空间
  - 程序设计时，易于区分是对内存操作还是对I/O端口操作
- 缺点：对I/O端口操作的指令类型少，操作不灵活
- 例子：8086/8088，分配给I/O端口的地址空间64K，0000H~0FFFFH，只能用IN和OUT指令对其进行读写操作

火龙果•整理
uml.org.cn

I/O控制器	I/O地址	中断向量
时钟	040-043	8
键盘	060-063	9
辅助RS-232	2F8-2FF	11
硬盘	320-32F	13
打印机	378-37F	15
单色显示	380-3BF	-
彩色显示	3DO-3DF	-
软盘	3FO-3F7	14
主RS-232	3F8-3FF	12

PC上的I/O控制器及其对应的I/O地址和中断向量

火龙果•整理
uml.org.cn

内存映像编址

- 分配给系统中所有端口的地址空间与内存的地址空间统一编址
- 把I/O端口看作一个存储单元，对I/O的读写操作等同于对存储器的操作
- 优点
  - 凡是可对存储器操作的指令都可对I/O端口操作
  - 不需要专门的I/O指令
  - I/O端口可占有较大的地址空间
- 缺点：占用内存空间

火龙果•整理
uml.org.cn

两者比较(1/3)

- I/O设备驱动程序：C语言 or 汇编语言？
- 对于内存映射I/O，不需要特殊的保护机制来阻止用户进程执行I/O操作。操作系统必须要做的全部事情只是避免把包含控制寄存器的那部分地址空间放入任何用户的虚拟地址空间之中
- 对于内存映射I/O，可以引用内存的每一条指令也可以引用控制寄存器。例如，如果存在一条指令TEST可以测试一个内存字是否为0，那么它也可以用来测试一个控制寄存器是否为0

```
LOOP：  TEST  PORT-4        //检测端口4是否为0
        BEQ  READY          //如果为0，转向READY
        BRANCH  LOOP        //否则，继续测试
READY：
```

火龙果•整理
uml.org.cn

两者比较(2/3)

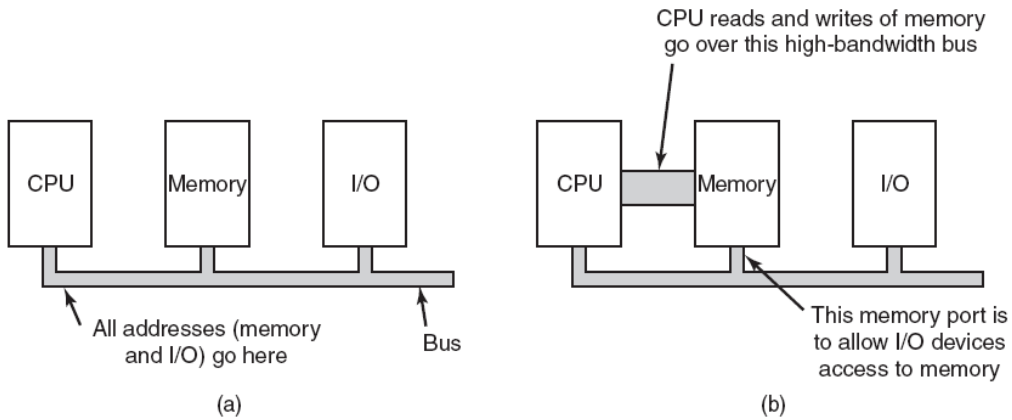
- 对一个设备控制寄存器不能进行高速缓存
- 考虑以下汇编代码循环，第一次引用PORT_4将导致它被高速缓存，随后的引用将只从高速缓存中取值并且不会再查询设备，之后当设备最终变为就绪时，软件将没有办法发现这一点，结果循环将永远进行下去
- 为避免这一情形，硬件必须针对每个页面具备选择性禁用高速缓存的能力，操作系统必须管理选择性高速缓存，所以这一特性为硬件和操作系统两者增添了额外的复杂性
- 其次，如果只存在一个地址空间，那么所有的内存模块和所有的I/O设备都必须检查所有的内存引用，以便了解由谁做出响应

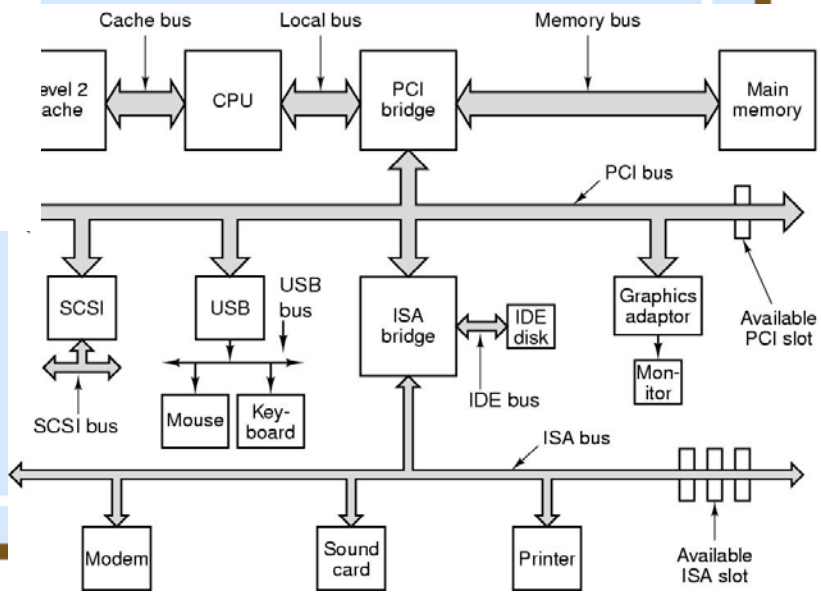
```
LOOP：TEST PORT-4        //检测端口4是否为0
      BEQ READY        //如果为0，转向READY
      BRANCH LOOP     //否则，继续测试
READY：
```

火龙果•整理
uml.org.cn

两者比较(3/3)

- 如果只存在一个地址空间，那么所有的内存模块和所有的I/O设备都必须检查所有的内存引用，以便了解由谁做出响应
- 如果计算机具有单一总线，那么让每个内存模块和I/O设备查看每个地址是简单易行的





火龙果•整理
uml.org.cn

4. I/O控制方式(1/3)

（1）直接I/O（轮询）

由CPU负责在机器内存与设备控制器数据寄存器之间进行数据传送

（2）中断驱动I/O

为了减少设备驱动程序不断地询问控制器状态寄存器的开销

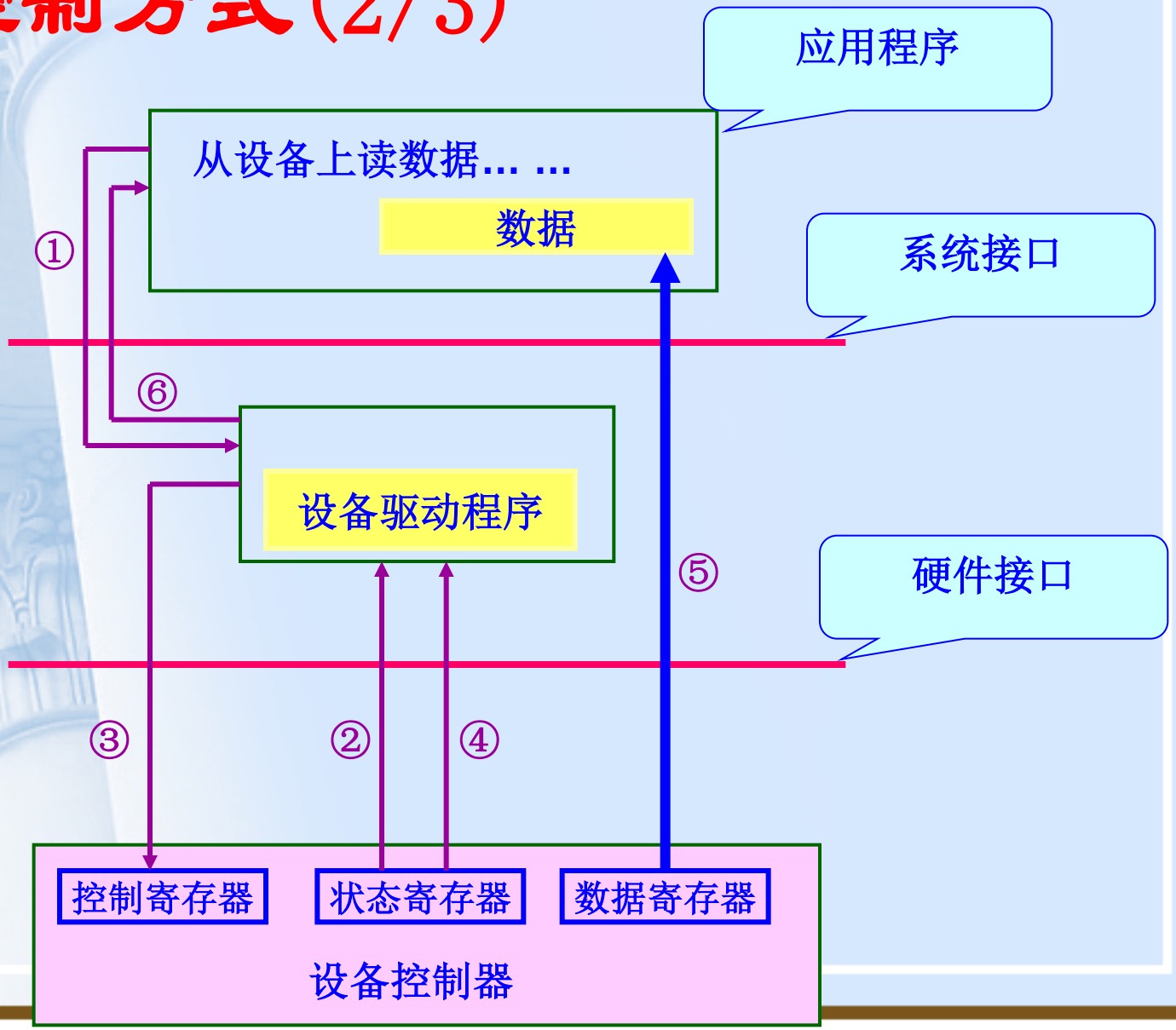
当I/O操作结束后，由设备控制器主动通知设备驱动程序

（3）DMA

（4）通道

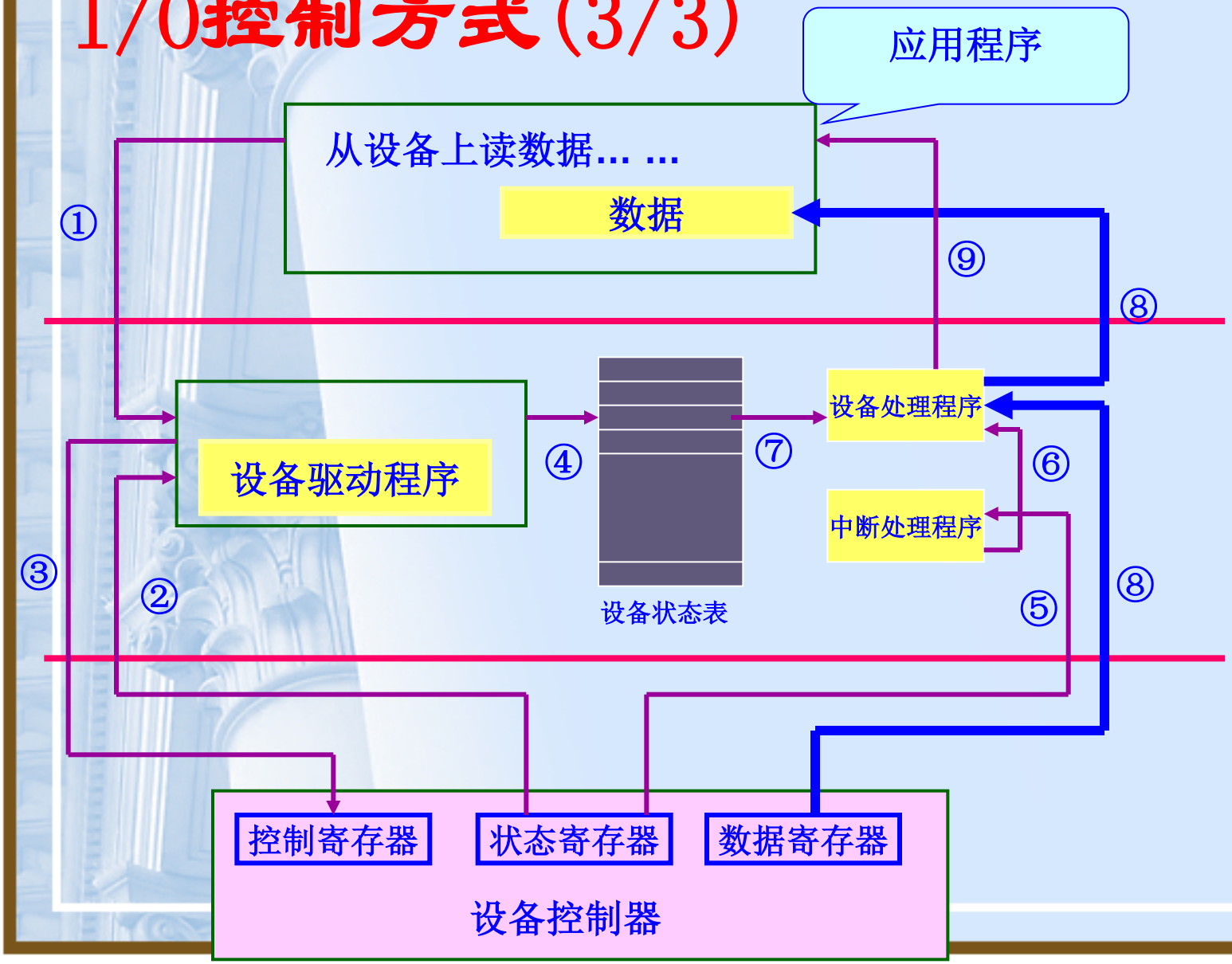
火龙果•整理
uml.org.cn

I/O控制方式(2/3)



火龙果•整理
uml.org.cn

I/O控制方式(3/3)



火龙果•整理
uml.org.cn

三、I/O软件的组成

I／O软件设计的基本思想——分层

- 把I/O软件组织成多个层次
- 较低层考虑硬件的特性，并向较高层软件提供接口
- 较高层不依赖于硬件，并向用户提供一个友好的、清晰的、简单的、功能更强的接口

火龙果•整理
uml.org.cn

设备独立性（设备无关性）

用户编写的程序可以访问任意I/O设备，无需事先指定设备

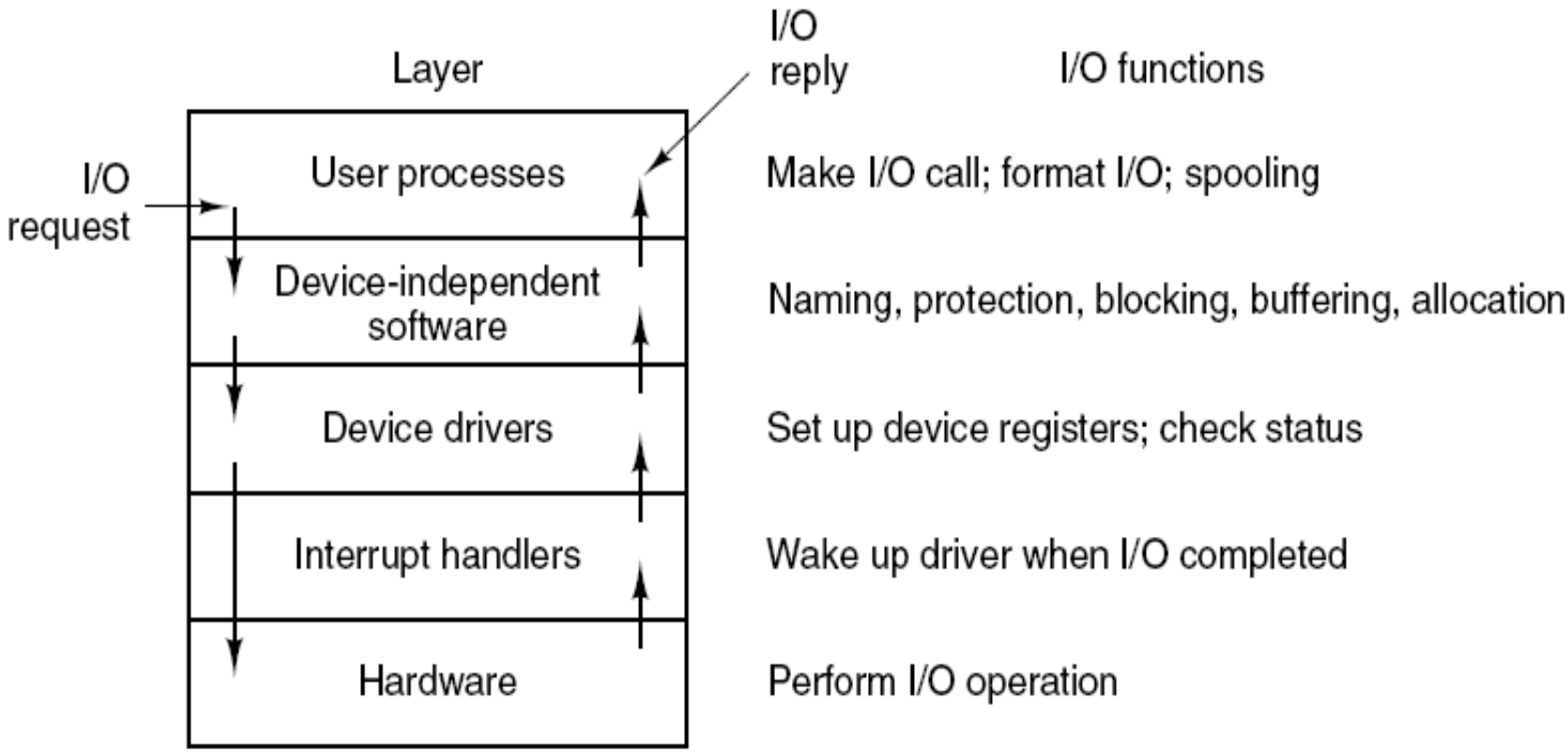
好处：
设备分配时的灵活性
易于实现I/O重定向

从用户角度：用户在编制程序时，使用逻辑设备名，由系统实现从逻辑设备到物理设备（实际设备）的转换，并实施I/O操作

从系统角度：设计并实现I／O软件时，除了直接与设备打交道的低层软件之外，其他部分的软件不依赖于硬件

火龙果•整理
uml.org.cn

1. I／O软件的层次



火龙果•整理
uml.org.cn

I/O软件各层的功能

（1）用户进程层执行输入输出系统调用，对I／0数据进行格式化，为假脱机输入／输出作准备

（2）独立于设备的软件实现设备的命名、设备的保护、成块处理、缓冲技术和设备分配

（3）设备驱动程序设置设备寄存器、检查设备的执行状态

（4）中断处理程序负责I／0完成时，唤醒设备驱动程序进程，进行中断处理

（5）硬件层实现物理I／0的操作

火龙果•整理
uml.org.cn

2. 设备驱动程序

- 与设备密切相关的代码放在设备驱动程序中，每个设备驱动程序处理一种设备类型
- 每一个控制器都设有一个或多个设备寄存器，用来存放向设备发送的命令和参数。设备驱动程序负责释放这些命令，并监督它们正确执行
- 一般，设备驱动程序的任务是接收来自与设备无关的上层软件的抽象请求，并执行这个请求
- 在设备驱动程序的进程释放一条或多条命令后，系统有两种处理方式，多数情况下，执行设备驱动程序的进程必须等待命令完成，这样，在命令开始执行后，它阻塞自己，直到中断处理时将它解除阻塞为止。而在其它情况下，命令执行不必延迟就很快完成

火龙果•整理
uml.org.cn

3. 设备独立的软件

设备驱动程序的统一接口
缓冲(Buffering)
错误报告
分配与释放专用设备
提供与设备无关的块大小

- 虽然I／O软件中一部分是设备特定的，但大部分软件是与设备无关的
- 设备驱动程序与设备独立软件之间的确切界限是依赖于具体系统的

火龙果•整理
uml.org.cn

4. 用户空间的I／O软件

尽管大部分I／O软件都包含在操作系统中，但仍有一小部分是由与用户程序连接在一起的库过程，甚至完全由运行于核外的程序构成。系统调用，包括I／O系统调用，通常由库过程实现

这些过程所做的工作只是将系统调用时所用的参数放在合适的位置，由其它的I／O过程实际实现真正的操作

火龙果•整理
uml.org.cn

四、设备相关技术

1. Spooling（虚拟设备）技术

一个虚拟设备

一个资源转换技术

（用空间，如输入，输出等换取CPU时间）

解决问题：

在进程所需物理设备不存在或被占用时使用该设备

火龙果•整理
uml.org.cn

打印机的SPOOLing值班进程

假若进程打开打印机特殊文件后几小时内无所事事，其他进程什么都打印不了！

解决方案：

- 创建值班（精灵）进程（daemon）、SPOOLing目录
- 进程首先生成要打印的文件，放入SPOOLing目录
- 值班进程：唯一获准使用打印机特殊文件的进程
  用以打印SPOOLing目录里的文件
- 通过禁止对特殊文件的直接使用→提高使用效率

火龙果•整理
uml.org.cn

网络的SPOOLing值班进程

SPOOLing技术今天仍被广泛使用

- 网络文件传送

  先把文件送到网络SPOOLing目录，然后网络值班进程把它取出并传递到目标地址

- Internet电子邮件系统

  为了寄邮，调用电子邮件程序

  待发信存在SPOOLing中供以后传输

- 注意：SPOOLing只提高设备利用率，缩短用户程序执行时间，并不提高CPU利用率

火龙果•整理
uml.org.cn

2. 通道技术

（1）定义：通道是独立于CPU的专门负责数据输入/输出传输工作的处理机，对外部设备实现统一管理，代替CPU对输入/输出操作进行控制，从而使输入，输出操作可与CPU并行操作。

可以执行通道程序

（2）引入通道的目的

为了使CPU从I/O事务中解脱出来，同时为了提高CPU与设备，设备与设备之间的并行工作能力

火龙果•整理
uml.org.cn

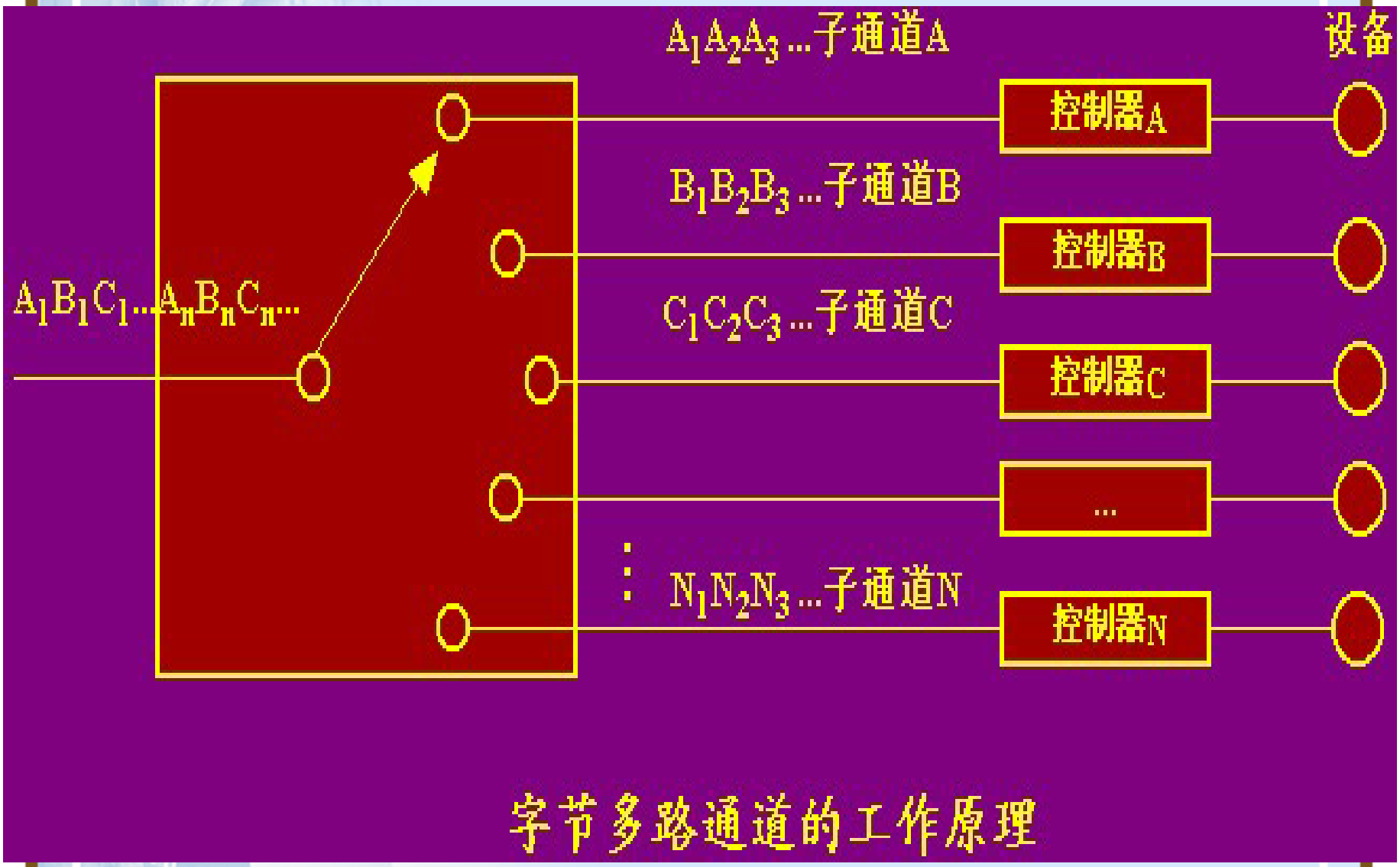
（3）通道的分类(1/5)

- 字节多路通道

字节多路通道以字节为单位传输信息，它可以分时地执行多个通道程序。当一个通道程序控制某台设备传送一个字节后，通道硬件就控制转去执行另一个通道程序，控制另一台设备传送信息

主要连接以字节为单位的低速I/O设备，如打印机，终端以字节为单位交叉传输，当一台传送一个字节后，立即转去为另一台传送字节

火龙果•整理
uml.org.cn



字节多路通道的工作原理

火龙果•整理
uml.org.cn

分类(3/5)

- 选择通道

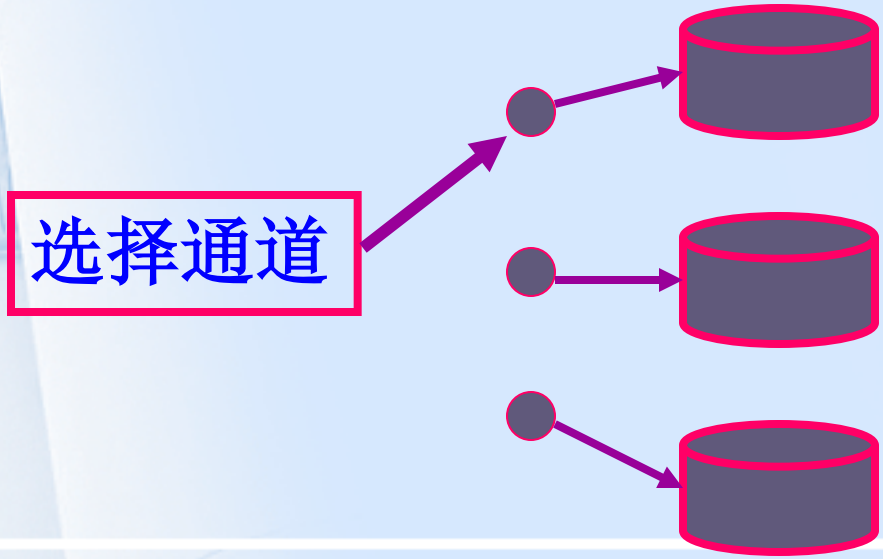
选择通道是以成组方式工作的，即每次传送一批数据，故传送速度很高。选择通道在一段时间内只能执行一个通道程序，只允许一台设备进行数据传输

火龙果•整理
uml.org.cn

分类(4/5)

当这台设备数据传输完成后，再选择与通道连接的另一台设备，执行相应的通道程序

主要连接磁盘，磁带等高速I/O设备



火龙果•整理
uml.org.cn

分类(5/5)

- 成组多路通道

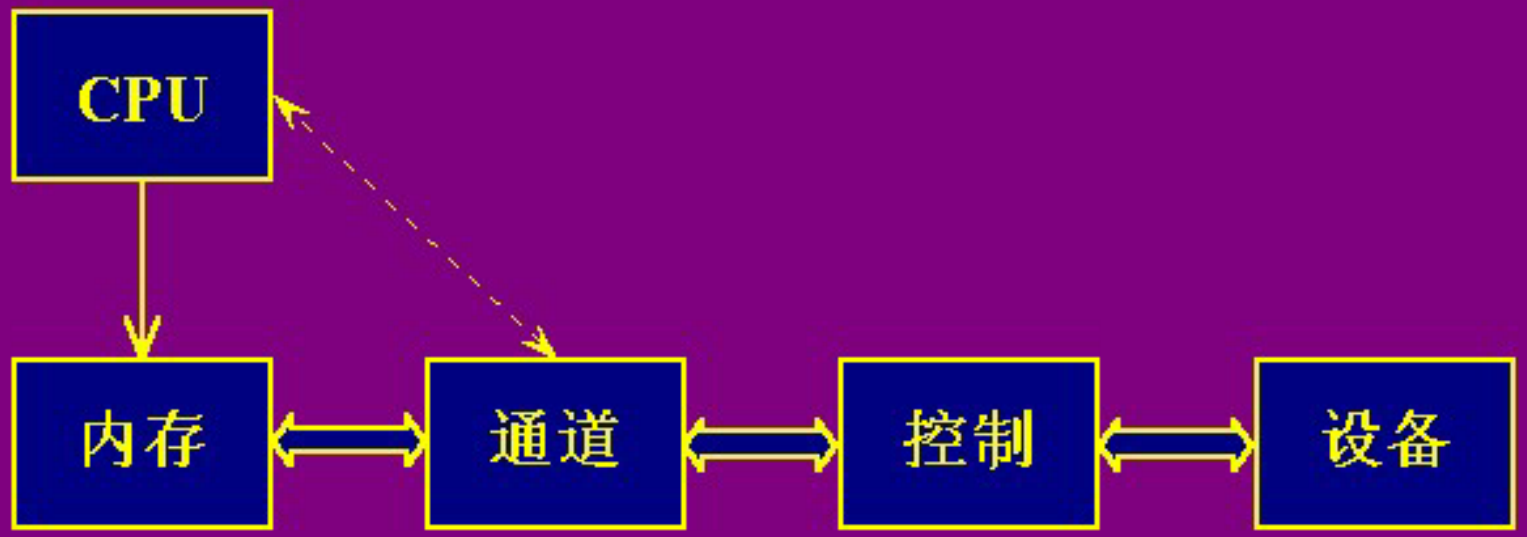
它结合了选择通道传送速度高和字节多路通道能进行分时并行操作的优点。它先为一台设备执行一条通道指令，然后自动转接，为另一台设备执行一条通道指令

主要连接高速设备

这样，对于连接多台磁盘机的数组多路通道，它可以启动它们同时执行移臂定位操作，然后，按序交叉地传输一批批数据。数据多路通道实际上是对通道程序采用多道程序设计的硬件实现

火龙果•整理
uml.org.cn

(4) 硬件连接结构



火龙果•整理
uml.org.cn

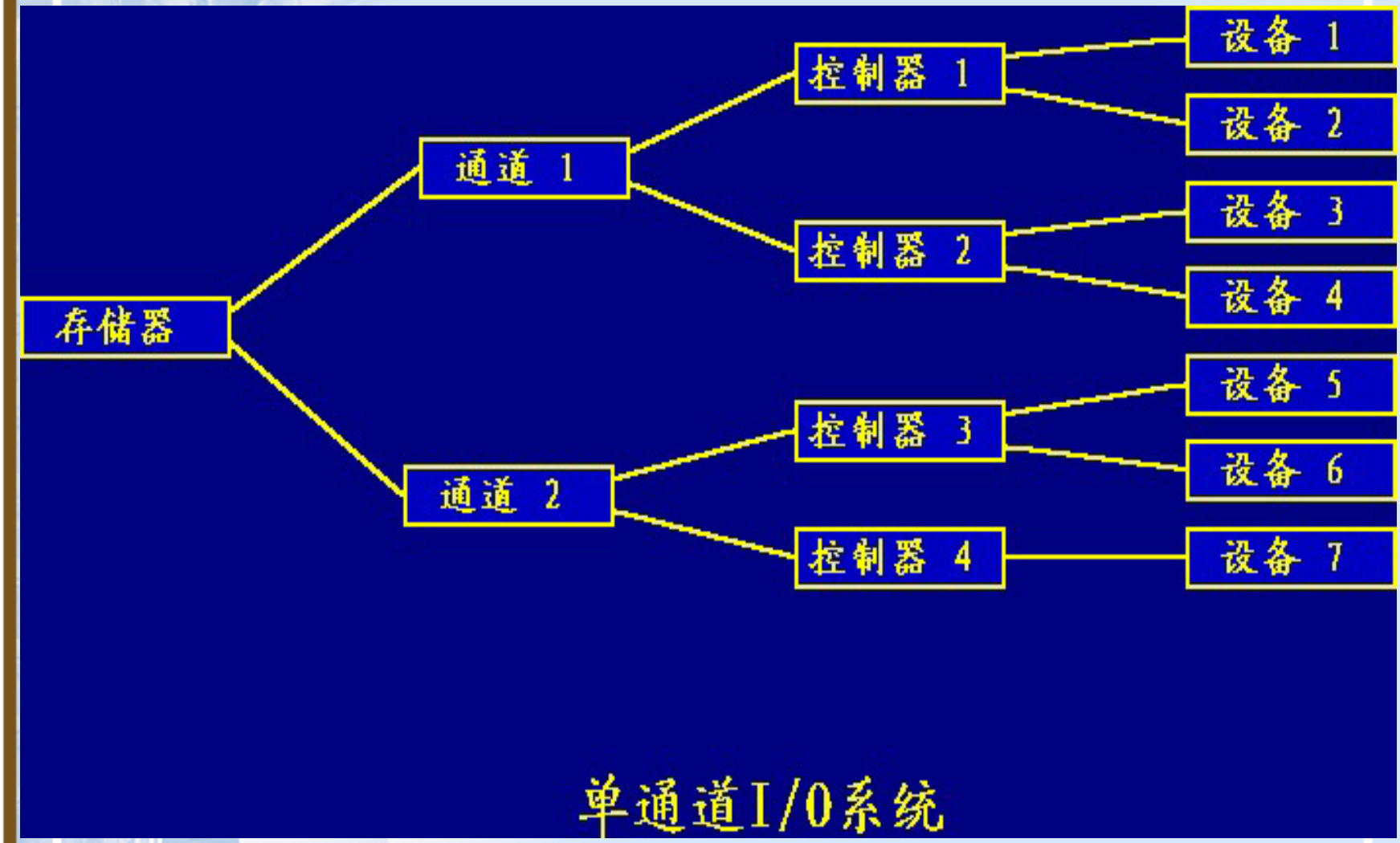
通道连接

通道：执行通道程序，向控制器发出命令，并具有向CPU发中断信号的功能。 一旦CPU发出指令，启动通道，则通道独立于CPU工作。一个通道可连接多个控制器，一个控制器可连接多个设备，形成树形交叉连接

主要目的是启动外设时：

- 提高了控制器效率
- 提高可靠性
- 提高并行度

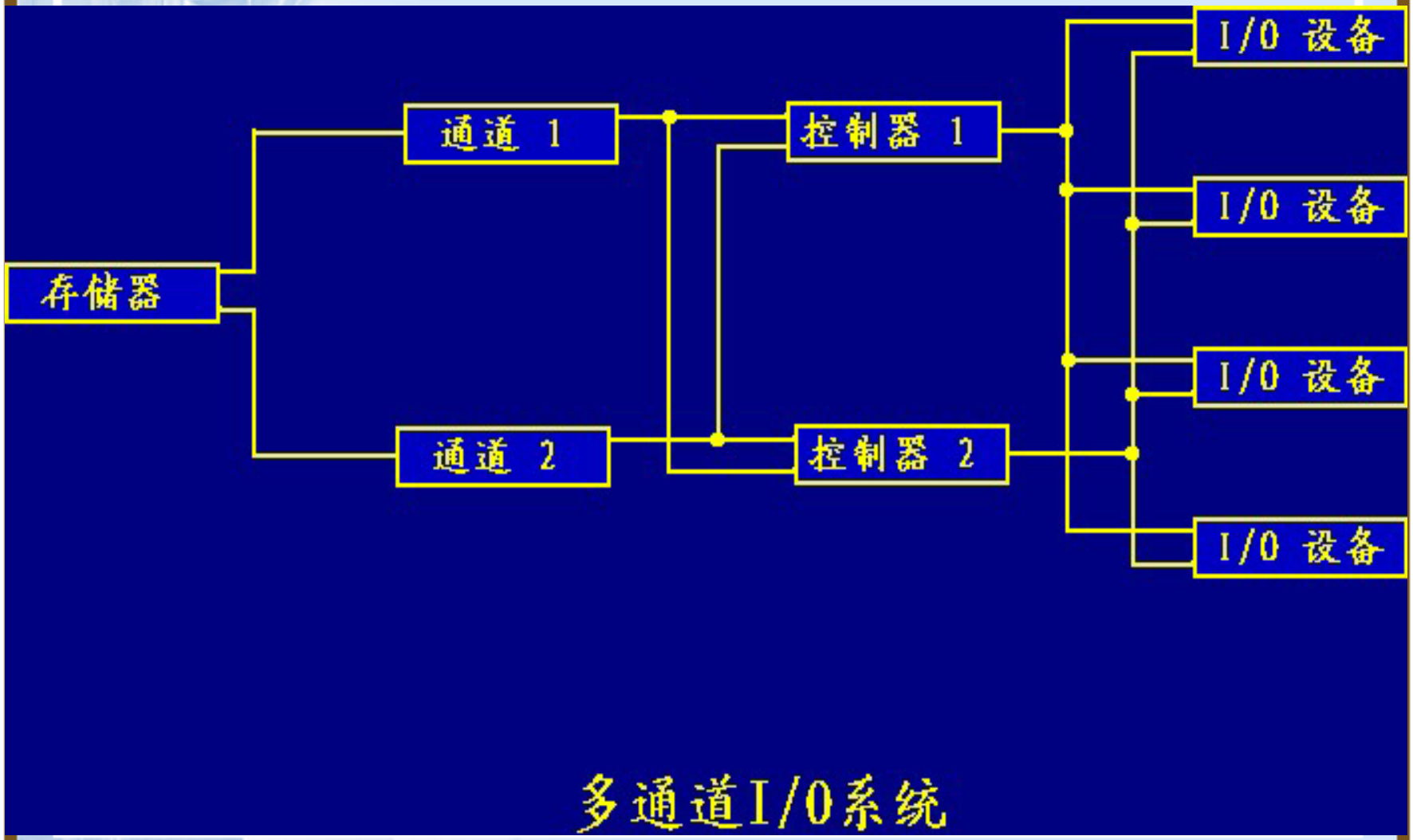
火龙果•整理
uml.org.cn



单通道I/O系统

火龙果•整理
uml.org.cn

交叉连接



多通道I/O系统

火龙果•整理
uml.org.cn

(5)通道工作原理(1/4)

通道相当于一个功能简单的处理机，包含通道指令（空操作，读操作，写操作，控制，转移操作），并可执行用这些指令编写的通道程序

- 通道运算控制部件

通道地址字 **CAW**：记录通道程序在内存中的地址

通道命令字 **CCW**： 保存正在执行的通道指令

通道状态字 **CSW**： 存放通道执行后的返回结果

通道数据字 **CDW**：存放传输数据

通道和**CPU**共用内存，通过周期窃取方式取得

火龙果•整理
uml.org.cn

通道工作原理(2/4)

- 通道命令及格式

用于I/O操作的命令主要有两种：

I/O指令：启动通道程序

通道命令：对I/O操作进行控制

读、反读、写、测试设备状态的数据传输命令、用于设备控制的命令（磁带反绕、换页）、实现通道程序内部控制的转移命令

命令格式一般包括：操作码、数据传输内存地址、特征位、计数器

例子：编制一个通道程序，从磁带机上读入200字节的信息，送入内存（1000）$_{16}$ 开始的单元。

通道命令码：“07”反绕

“02”读

CCW1 X “07” * X “40” 1；

CCW2 X “02” 1000 X “00” 200；

火龙果•整理
uml.org.cn

通道工作原理(3/4)

- 工作原理

CPU：执行用户程序，当遇到I/O请求时，可根据该请求生成通道程序放入内存（也可事先编好放入内存），并将该通道程序的首地址放入CAW中；之后执行“启动I/O”指令，启动通道工作

通道：接收到“启动I/O”指令后，从CAW中取出通道程序的首地址，并根据首地址取出第一条指令放入CCW中，同时向CPU发回答信号，使CPU可继续执行其他程序，而通道则开始执行通道程序，完成传输工作

火龙果•整理
uml.org.cn

通道工作原理(4/4)

（通道程序完成实际I/O，启动I/O设备，执行完毕后,如果还有下一条指令，则继续执行，否则表示传输完成）

当通道传输完成最后一条指令时，向CPU发I/O中断，并且通道停止工作。CPU接收中断信号，从CSW中取得有关信息，决定下一步做什么

火龙果•整理
uml.org.cn

通道的发展

- 新的通道思想综合了许多新的技术
- 在个人计算机中，芯片组中专门I/O处理的芯片，称为IOP（IO Processor），发挥通道的作用
- IBM 390 中，沿用了输入输出通道概念

  IBM于1998年推出光纤通道技术（称为FICON），可通过 FICON 连接多达127个大容量I/O设备

  光纤通道技术具有数据传输速率高、传输距离远，可简化大型存储系统设计的优点

  在大容量高速存储，如大型数据库、多媒体、数字影像等应用领域，有广泛前景

火龙果•整理
uml.org.cn

3. DMA技术(1/2)

不用DMA时，磁盘如何读：

首先，控制器从磁盘驱动器串行地一位一位地读一个块，直到将整块信息放入控制器的内部缓冲区中

其次，它做和校验计算，以核实没有读错误发生

然后控制器产生一个中断。CPU响应中断，控制转给操作系统。当操作系统开始运行时，它重复地从控制器缓冲区中一次一个字节或一个字地读这个磁盘块的信息，并将其送入内存中

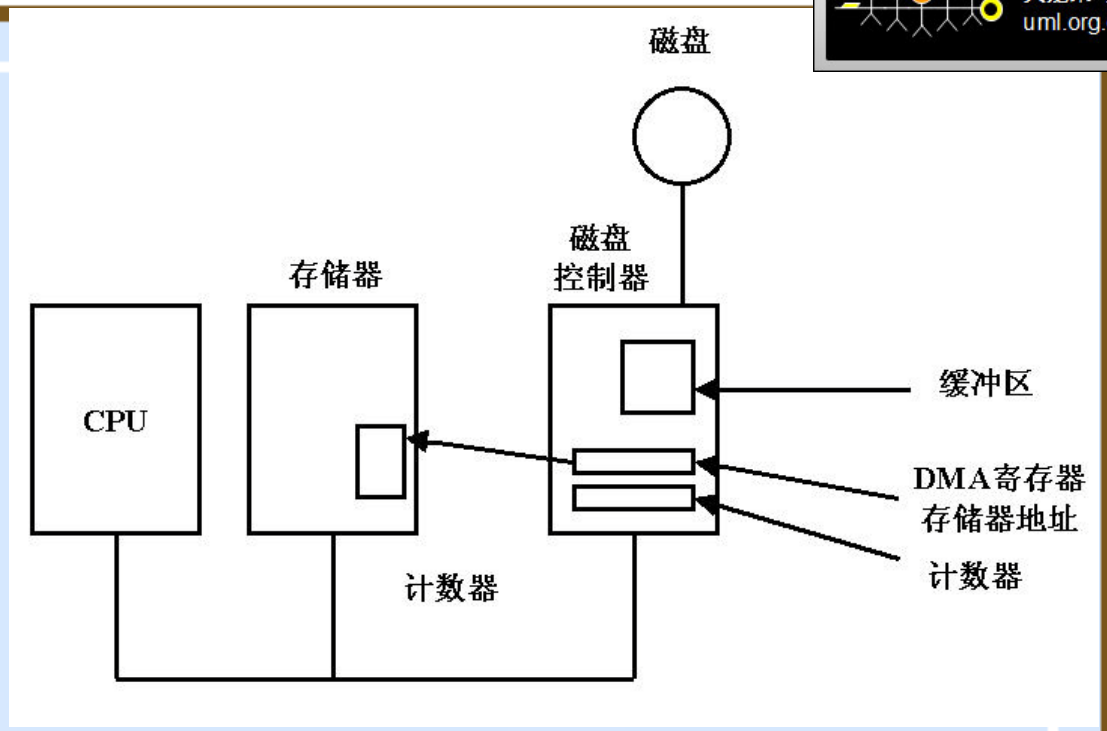
火龙果•整理
uml.org.cn

DMA技术(2/2)

- 数据在内存与I/O设备间的直接成块传送
- CPU在开始时向设备发“传送一块”命令，结束时进行相应处理，实际操作由DMA硬件直接完成
- DMA的功能可以以独立的DMA部件在系统I/O总线上完成，也可整合到I/O部件中完成
- DMA通过系统总线代替CPU管理数据的存入或取出
  - 当CPU不需要系统总线时可以使用总线
  - DMA可以强迫CPU暂时延迟其他操作，获取一个总线周期（周期窃取）
- 不需要CPU介入，因而减轻了CPU负担

火龙果•整理
uml.org.cn

DMA工作示例（以硬盘为例）



CPU提供

- 待读取块的磁盘地址
- 目标存储地址
- 待读取字节数

整块数据读进缓冲区

核准校验

控制器按照指定存储器地址，把第一个字节送入内存

然后，按指定字节数进行数据传送

每当传送一个字节后，字节计数器值减1，直到字节计数器等于0

此时，控制器引发中断，通知操作系统，操作完成

火龙果•整理
uml.org.cn

4. 缓冲技术(1/5)

（1）缓冲技术的引入

最早引入：CPU与I/O设备之间

凡是数据到达和离去速度不匹配的地方均可采用缓冲技术

目的：

- 缓解CPU与I/O设备之间速度不匹配的矛盾
- 提高CPU与I/O设备之间的并行性
- 减少了I/O设备对CPU的中断请求次数，放宽CPU对中断响应时间的要求

火龙果•整理
uml.org.cn

缓冲技术(2/5)

（2）缓冲区设置

硬缓冲：由硬件寄存器实现（例如：设备中设置的缓冲区）

软缓冲：在内存中开辟一个空间，用作缓冲区

（3）缓冲区管理

单缓冲：当用户进程发出I/O请求时，操作系统在内存的系统空间为该操作分配一个缓冲区，可以实现提前读和延迟写

双缓冲：可以实现用户数据区—缓冲区之间交换数据和缓冲区—外设之间交换数据的并行

火龙果•整理
uml.org.cn

缓冲技术(3/5)

缓冲池：

又称循环缓冲

多个缓冲区连接起来统一管理

引入系统缓冲池，采用有界缓冲区的生产者/消费者模型对缓冲池中的缓冲区进行循环使用

常采用多缓冲管理

缓冲区结合提前读和延迟写技术对具有重复性及阵发性I/O进程、提高I/O速度很有帮助

火龙果•整理
uml.org.cn

缓冲技术(4/5)

（4）例子

终端输入软件中的键盘驱动程序

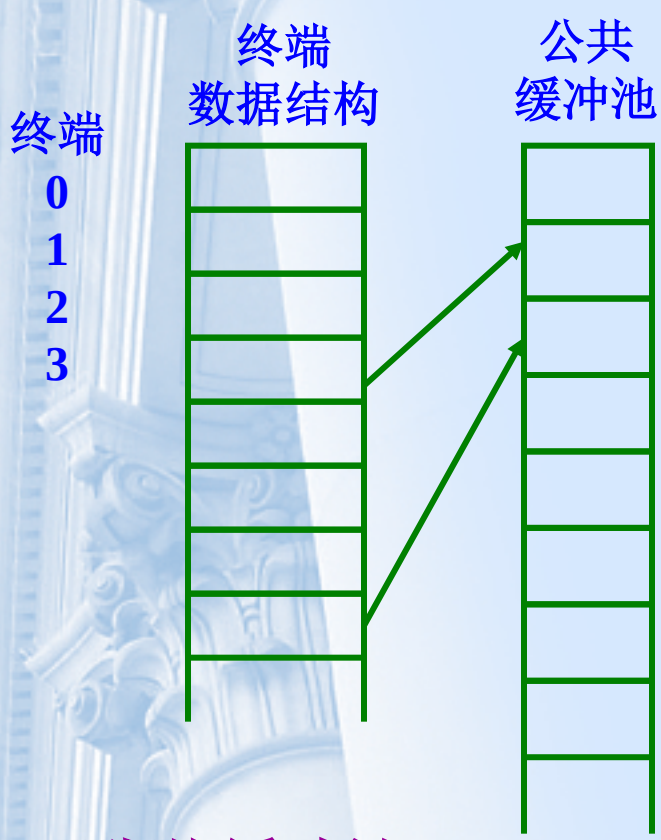
任务之一：收集字符

两种常见的字符缓冲方法：

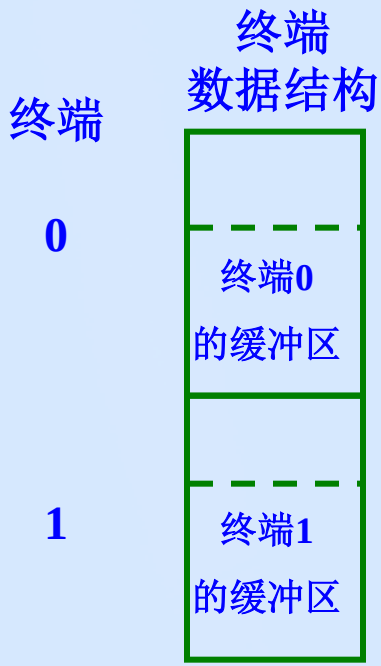
- 公共缓冲池（驱动程序中）
- 终端数据结构缓冲

火龙果•整理
uml.org.cn

缓冲技术(5/5)



公共缓冲池



终端固定缓冲区

火龙果•整理
uml.org.cn

UNIX缓冲技术(1/8)

- 采用缓冲池技术，来平滑和加快文件信息从内存到磁盘的传输
- 并且，充分利用以前从磁盘读入已传入用户区、但仍在缓冲区的数据

  即：当从磁盘上读数据时，如果数据已经在缓冲区中，则系统直接从缓冲区中读出，而不必从盘上读；当数据不在缓冲区时，系统首先把数据从磁盘传送到缓冲区，再从缓冲区读出

  好处：尽可能减少磁盘I/O的次数，提高系统运行的速度

火龙果•整理
uml.org.cn

UNIX缓冲技术(2/8)

UNIX System V

- 由200个缓冲区组成的缓冲池
- 每个缓冲区的长度可以是512字节或1024字节
- 每个缓冲区由两部分组成：
  缓冲数据区：存放数据的区域
  缓冲控制块或缓冲首部：用于控制的区域

系统通过缓冲控制块来实现对缓冲区的管理

火龙果•整理
uml.org.cn

UNIX缓冲技术(3/8)

说明：

- 逻辑设备号和盘块号分别标志出文件系统和数据所在的盘块号，是缓冲区的唯一标志
- 状态项指明该缓冲区当前的状态：忙／闲、上锁／开锁、是否延迟写、数据有效性等
- 两组指针（av和b）用于对缓冲池的分配管理

设备号
盘块号
状态
指向缓冲数据区的指针
指向空闲队列的后继指针b-back
指向空闲队列的前驱指针b-forw
指向缓冲队列的后继指针av-back
指向缓冲队列的前驱指针av-forw

火龙果•整理
uml.org.cn

UNIX缓冲技术(4/8)

缓冲池结构：

- 空闲缓冲区队列(空闲av链)

  连接高速缓冲中可被重新分配的缓冲区

  队列头部为bfreelist

  系统初始化时，所有缓冲区按序号大小挂在空闲av链上，当申请一个缓冲区时，从空闲av链队首取下一个缓冲区，而释放一个缓冲区时将其挂入空闲av链队尾

火龙果•整理
uml.org.cn

UNIX缓冲技术(5/8)

- 设备缓冲队列（设备b链）
  链接所有分配给各类设备使用的缓冲区
  按照散列方式组织

buf用b双向链，可以有64个队列，每个队列头部有头标

设备为b_dev上的逻辑块b在散列队列的头标为：

$$i=(b_dev + b) \bmod 64$$

火龙果•整理
uml.org.cn

UNIX缓冲技术(6/8)

每个buf同时在av链和b链：

- 开始：在空闲av链(缓冲区未被使用时)
- 开始IO请求：在设备IO请求队列和设备b链
- IO完成：在空闲av链和设备b链

火龙果•整理
uml.org.cn

UNIX缓冲技术(7/8)

当进程想从指定的盘块读取数据时，系统根据盘块号从设备b链(散列队列)中查找，如找到缓冲区，则将该缓冲区状态标记为“忙”，并从空闲av队列中取下，接着完成从缓冲区到内存用户区的数据传送。如果在设备b链中未找到时，则从空闲av链队首摘取一个缓冲区，插入设备I/O请求队列；并从原设备b链中取下，插入由读入信息盘块号确定的新的设备b链中。当数据从磁盘块读入到缓冲区后，缓冲区从设备I/O请求队列取下。当系统完成从缓冲区到内存用户区的数据传送后，要把缓冲区释放，链入空闲av链队尾

火龙果•整理
uml.org.cn

UNIX缓冲技术(8/8)

当数据从磁盘块读入到缓冲区，并传送到内存用户区后，该缓冲区一直保留在原设备b链中，即它的数据一直有效。如它又要被使用，则又要从空闲av链中取下，使用完后插入到空闲av链队尾。如它一直未使用，则该缓冲区从空闲av链队尾慢慢升到队首，最后被重新分配，旧的盘块数据才被置换

系统对缓冲区的分配是采用近似LRU算法

火龙果•整理
uml.org.cn

五、设备管理

1. 设备的分配与回收(1/3)

当某进程向系统提出I/O请求时，设备分配程序按一定策略分配设备、控制器和通道，形成一条数据传输通路，以供主机和设备间信息交换

火龙果•整理
uml.org.cn

设备分配与回收(2/3)

（1）设备管理有关的数据结构

- 描述设备、控制器等部件的表格：系统中常常为每一个部件、每一台设备分别设置一张表格，常称为设备表或部件控制块。这类表格具体描述设备的类型、标识符、状态，以及当前使用者的进程标识符等
- 建立同类资源的队列：系统为了方便对I/O设备的分配管理，通常在设备表的基础上通过指针将相同物理属性的设备连成队列（称设备队列）
- 面向进程I/O请求的动态数据结构：每当进程发出I/O请求时，系统建立一张表格（称I/O请求包），将此次I/O请求的参数填入表中，同时也将该I/O有关的系统缓冲区地址等信息填入表中。I/O请求包随着I/O的完成而被删除
- 建立I/O队列：如请求包队列

火龙果•整理
uml.org.cn

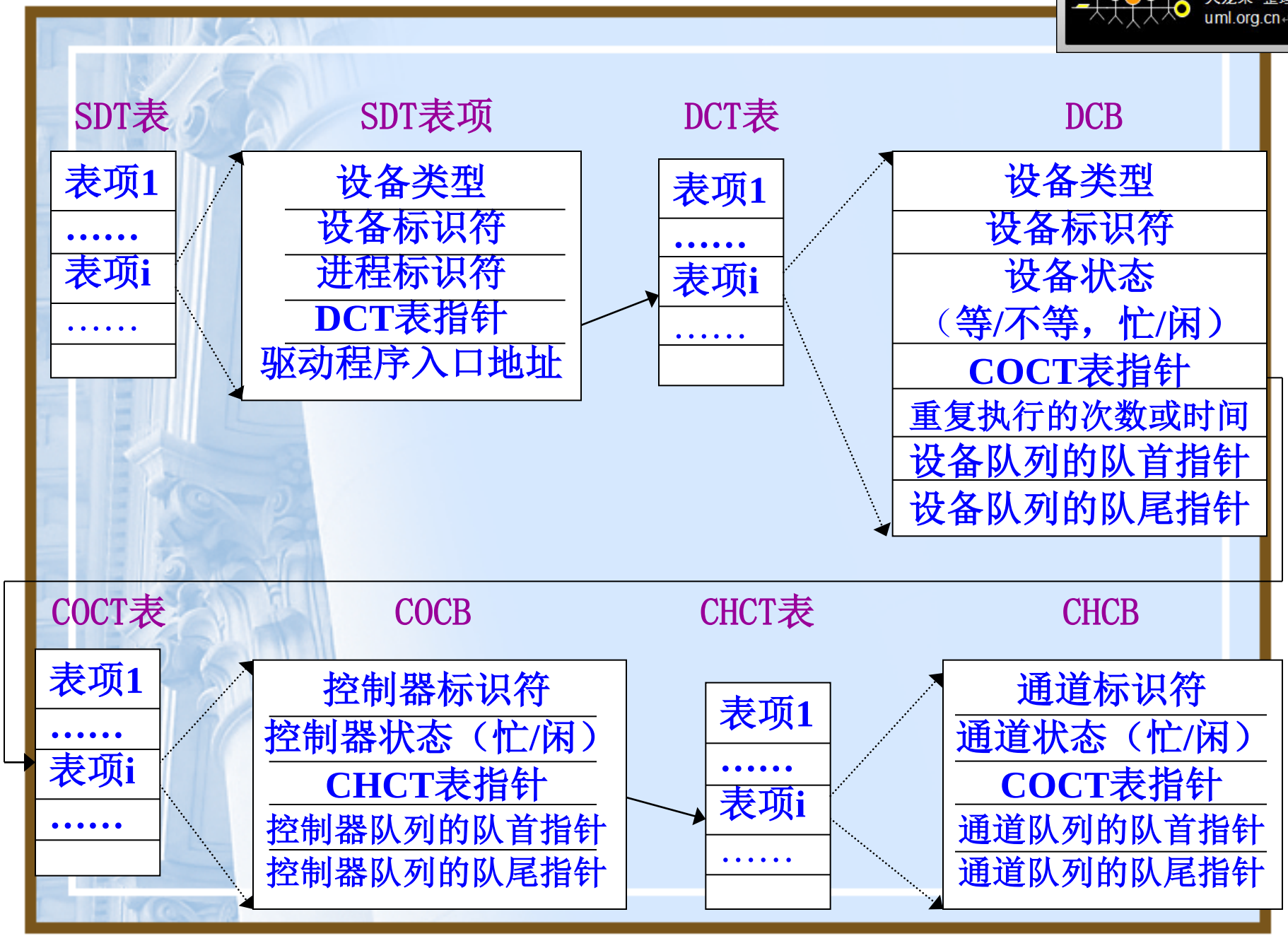
设备分配与回收(3/3)

（2）设备有关的数据结构

- 设备控制块DCB（设备控制表DCT）
- 控制器控制块COCB（控制器控制表COCT）
- 通道控制块CHCB（通道控制表CHCT）
- 系统设备表SDT

根据用户请求的I/O设备的逻辑名，查找逻辑设备和物理设备的映射表；以物理设备为索引，查找SDT，找到该设备所连接的DCT；继续查找与该设备连接的COCT和CHCT，就找到了一条通路

火龙果•整理
uml.org.cn



火龙果•整理
uml.org.cn

2. 设备分配策略(1/3)

- 独占设备的分配

  在申请设备时，如果设备空闲，就将其独占，不再允许其他进程申请使用，一直等到该设备被释放，才允许被其他进程申请使用

  考虑效率问题，并避免由于不合理的分配策略造成死锁

静态分配：

在进程运行前，完成设备分配；运行结束时，收回设备

缺点：设备利用率低

动态分配：

在进程运行过程中，当用户提出设备要求时，进行分配，一旦停止使用立即收回

优点：效率好；缺点：分配策略不好时，产生死锁

火龙果•整理
uml.org.cn

设备分配策略(2/3)

- 分时式共享设备分配

  所谓分时式共享就是以一次I/O为单位分时使用设备，不同进程的I/O操作请求以排队方式分时地占用设备进行I/O

  由于同时有多个进程同时访问，且访问频繁，就会影响整个设备使用效率，影响系统效率。因此要考虑多个访问请求到达时服务的顺序，使平均服务时间越短越好

火龙果•整理
uml.org.cn

设备分配策略(3/3)

- 以SPOOLing方式使用外设
  - SPOOLing 技术是在批处理操作系统时代引入的，即所谓假脱机输入/输出技术。把这种技术用于对设备的使用实质就是对输入／输出操作成批处理。
  - 例如:所有输出数据已经写到文件当中，并排到打印输出队列，打印进程申请占用打印机后，成批读出文件中数据，并送打印机打印出去

火龙果•整理
uml.org.cn

3. 设备驱动程序(1/3)

为了控制I/O传输，系统为每类设备编制设备驱动程序

任务：主要负责接收和分析从设备分配转来的信息，并根据设备分配的结果，结合具体物理设备特性完成以下具体工作

火龙果•整理
uml.org.cn

设备驱动程序(2/3)

(1) 接受来自上层、与设备无关软件的抽象I/O请求，将该请求排在请求队列末尾，检查I/O请求的合法性

(2)执行特定的缓冲区策略：根据请求传输的数据量，组织I/O缓冲队列，利用I/O缓冲对数据进行加工，包括数据格式处理和编码转换

(3)对各种可能的有关设备排队、挂起、唤醒等操作进行处理

(4)向有关的输入输出设备的各种控制器发出控制命令，并且监督它们的正确执行，进行必要的错误处理

(5)处理来自设备的中断

火龙果•整理
uml.org.cn

设备驱动程序(3/3)

设备驱动程序与外界的接口

（1）与操作系统的接口

为实现设备无关性，设备作为特殊文件处理。用户的I/O请求、对命令的合法性检查以及参数处理在文件系统中完成。在需要各种设备执行具体操作时，通过相应数据结构转入不同的设备驱动程序

（2）与系统引导的接口（初始化，包括分配数据结构，建立设备的请求队列）

（3）与设备的接口

火龙果•整理
uml.org.cn

设备驱动程序接口函数(1/2)

- **驱动程序初始化函数**：这个函数是为了使驱动程序其他函数能被上层正常调用，而做一些针对驱动程序本身的初始化工作，如向操作系统登记该驱动程序的接口函数，该初始化函数在系统启动时或驱动程序安装入内核时执行
- **驱动程序卸载函数**：是驱动程序初始化函数的逆过程，在支持驱动程序可动态加载卸载的系统中才需要
- **申请设备函数**：该函数申请一个驱动程序所管理的设备，按照设备特性进行独占式占用或者分时共享式占用，如果是独占式申请成功还应该对设备做初始化工作
- **释放设备函数**：是申请设备函数的逆过程

火龙果•整理
uml.org.cn

设备驱动程序接口函数(2/2)

- **I／O操作函数：**这个函数实现对设备的I／O。如果对独占型设备，则包含了启动I／O的指令；对分时共享型设备，该函数通常将I／O请求形成一个请求包，将其排到设备请求队列，如果请求队列空，则直接启动设备

- **中断处理函数：**这个函数在设备I／O完成时向CPU发中断，然后被调用。该函数对I／O完成做善后处理，一般是找到等待刚完成I／O请求的阻塞进程，将其就绪，使其能进一步做后续工作。如果存在I／O请求队列，则启动下一个I／O请求

火龙果•整理
uml.org.cn

4. 一种实现方案：I/O进程(1/3)

主要有三种处理方式：

(1) 每类设备一个I/O进程

(2) 整个系统一个I/O进程

(3) 为各类设备设置相应的设备处理程序供外部调用

I/O进程：专门处理系统中的I/O请求和I/O中断工作

火龙果•整理
uml.org.cn

一种实现方案：I/O进程(2/3)

（1）I/O请求的进入

用户程序：调用send将I/O请求发送给I/O进程；调用block将自己阻塞，直到I/O任务完成后被唤醒

系统：利用wakeup唤醒I/O进程，完成用户所要求的I/O处理

（2）I/O中断的进入

当I/O中断发生时，内核中的中断处理程序发一条消息给I/O进程，由I/O进程负责判断并处理中断

火龙果•整理
uml.org.cn

一种实现方案：I/O进程(3/3)

（3）I/O进程

是系统进程，一般赋予最高优先级。一旦被唤醒，它可以很快抢占处理机投入运行

I/O进程开始运行后，首先关闭中断，然后用receive去接收消息。两种情形：

- 没有消息，则开中断，将自己阻塞；
- 有消息，则判断消息（I/O请求或I/O中断）；

a. I/O请求

准备通道程序，发出启动I/O指令，继续判断有无消息

b. I/O中断，进一步判断正常或异常结束

正常：唤醒要求进行I/O操作的进程

异常：转入相应的错误处理程序

火龙果•整理
uml.org.cn

I/O的操作全过程

火龙果•整理
uml.org.cn

小结——I/O性能问题

使CPU利用率尽可能不被I/O降低

使CPU尽可能摆脱I/O

- 减少或缓解速度差距 → 缓冲技术
- 使CPU不等待I/O → 异步I/O
- 让CPU摆脱I/O操作 → DMA、通道

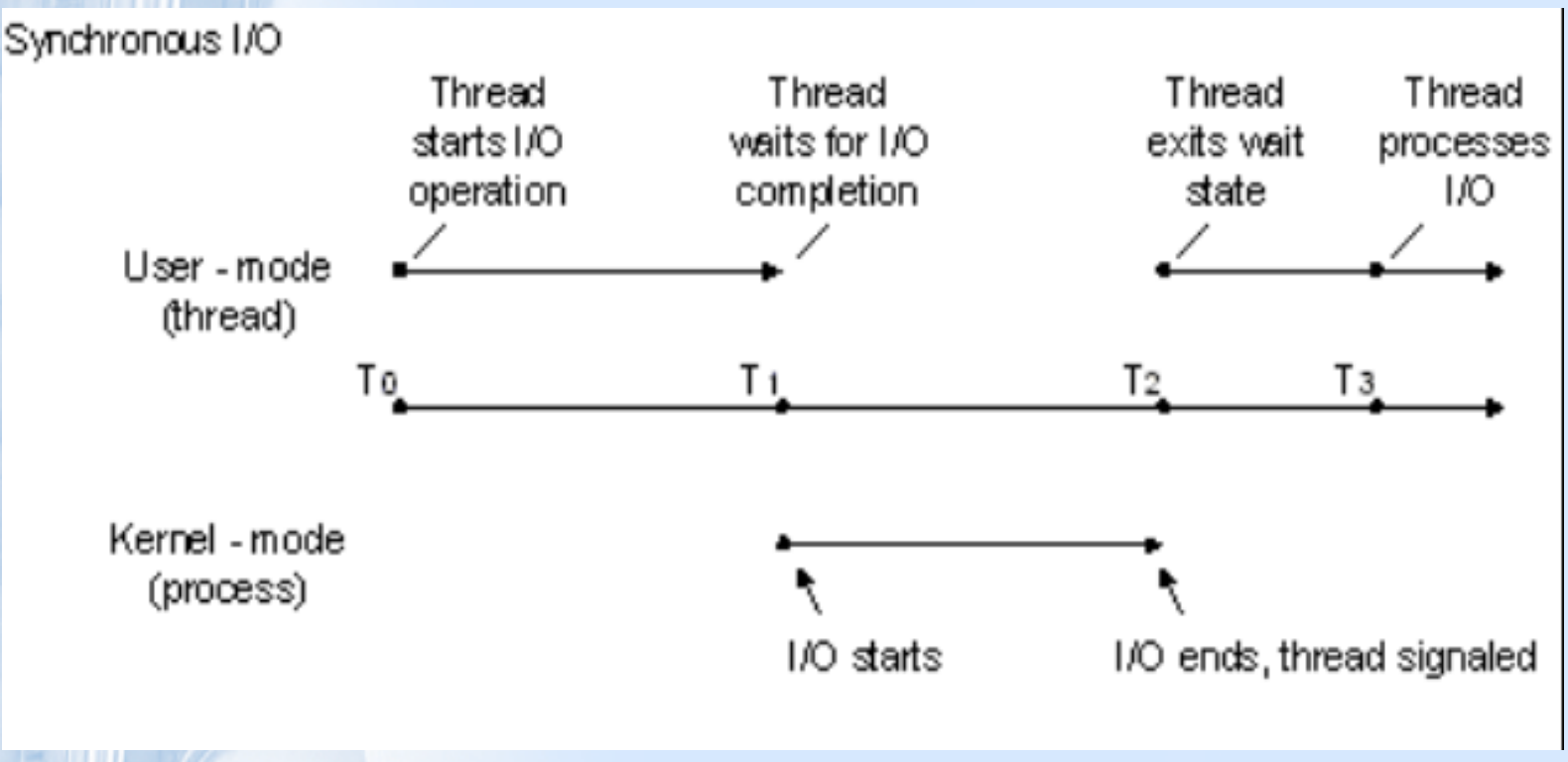
火龙果•整理
uml.org.cn

异步传输

基本思想：填充I/O操作间等待的CPU时间

火龙果•整理
uml.org.cn

同步传输I/O流程



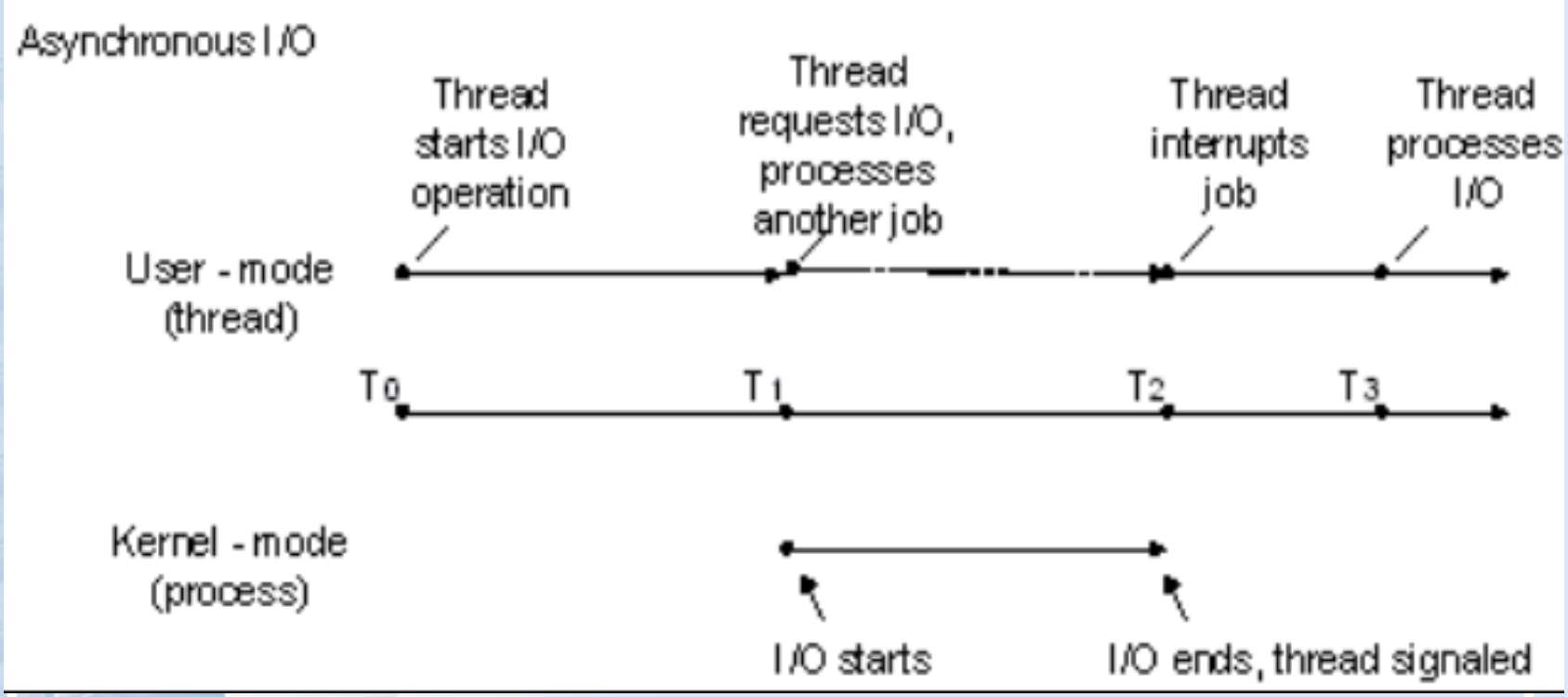
可见在I/O处理过程中，CPU处于空闲等待状态

而在处理数据的过程中，不能同时进行I/O操作

火龙果•整理
uml.org.cn

异步传输I/O的基本思想

- 数据处理和I/O操作不存在严格的时序关系，只有逻辑关系
- 只要数据就绪，对其的操作就可以进行
- 数据没有就绪的情况下，也可以做对其他数据进行处理



火龙果•整理
uml.org.cn

异步传输模式的实现

- 系统实现
  - 如上页图示，通过切换到其他线程保证CPU利用率
  - 对少量数据的I/O操作会引入切换的开销
- 用户实现
  - 将访问控制分成两段进行
  - 发出读取指令后继续做其他操作
  - 当需要用读入的数据的时候，再使用wait命令等待其完成
  - 不引入线程切换，减少开销

火龙果•整理
uml.org.cn

应用程序设计典型步骤

- 调用CreateEvent创建事件对象，返回值为该事件对象的句柄
- 建立一个OVERLAPPED结构的数据，并将其hEvent成员指定为上述事件对象句柄
- 调用CreateFile，以FILE_FLAG_OVERLAPPED方式建立或打开文件
- 调用ReadFile或WriteFile进行文件读写，其中的lpOverlapped指向上述OVERLAPPED结构，表示进行的是异步操作
- 应用程序进行其他后续操作，当要查询上述异步操作是否完成时，调用GetOverlappedresult

火龙果•整理
uml.org.cn

重点总结

- 基本概念
  设备独立性、SPOOLing技术、虚设备
- I/O软件的组成
- I/O硬件的组成
- 设备分配与回收、独占设备分配算法、共享设备分配问题
- 通道/DMA工作原理
- 缓冲技术
- I/O性能问题