操作系统

Operating system

胡燕

大连理工大学

内容纲要

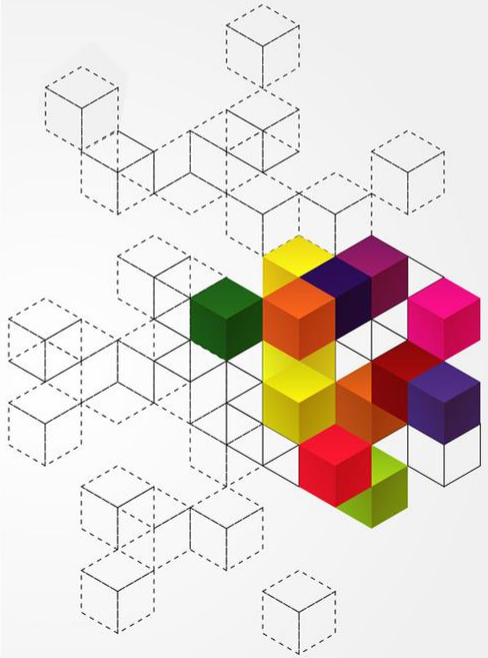
13.1 IO子系统概述

一、 IO子系统功能

二、 IO设备分类

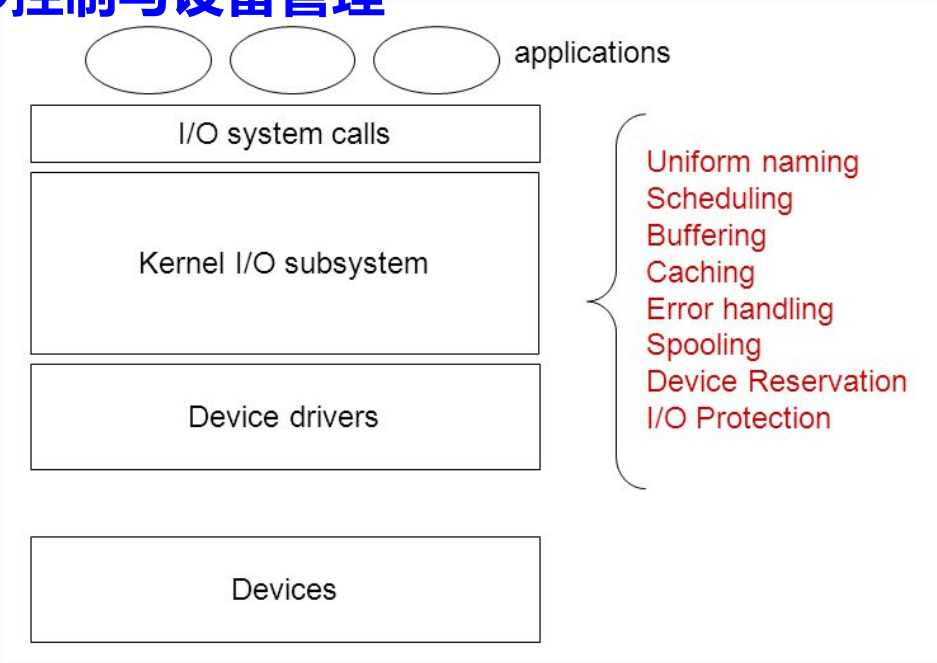
三、 设备控制器

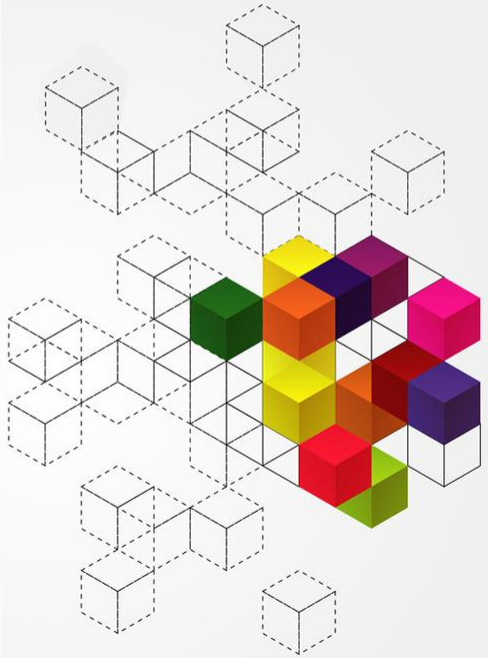
四、 总线及其分类



一、IO子系统功能

IO控制与设备管理





主要设计目标

-提高设备利用率

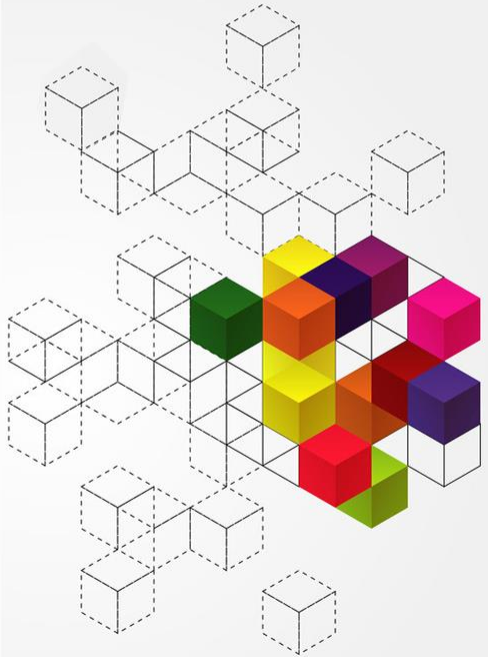
-统一界面，为用户提供逻辑设备(硬件抽象)

一、IO子系统功能

设备统一命名

程序猿利用系统调用打开IO设备时，通常使用的设备标识是（ ）。

A.主设备号
B.从设备号
C.逻辑设备名
D.物理设备名

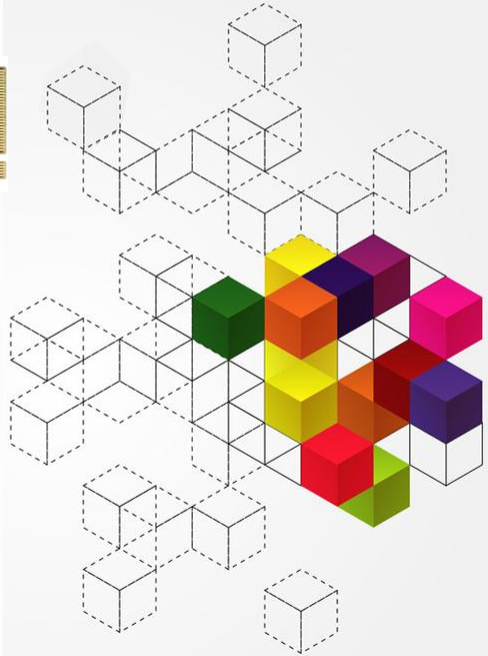


二、IO设备分类

IO设备是IO子系统的管理对象，类别繁多

分类准则

（1）传输速率



二、IO设备分类

IO设备是IO子系统的管理对象，类别繁多

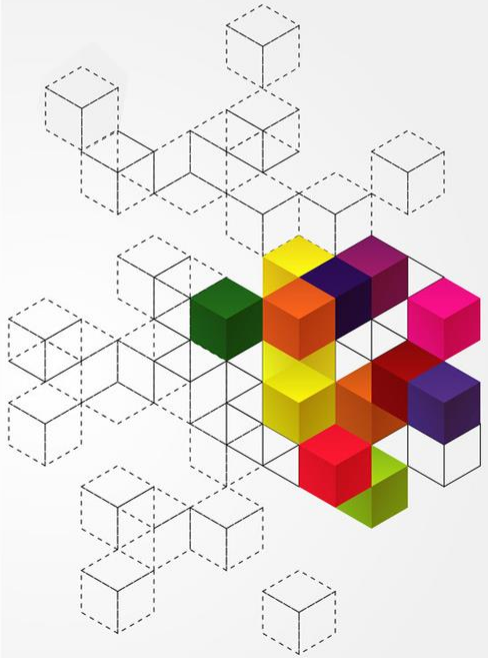
分类准则

（1）传输速率

（2）信息交换单位

字符设备

块设备



二、IO设备分类

IO设备是IO子系统的管理对象，类别繁多

分类准则

（1）传输速率

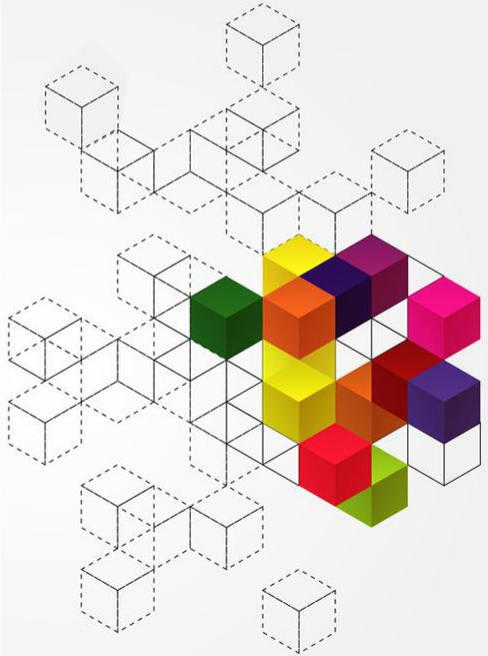
（2）信息交换单位

（3） 使用特性

人机交互类

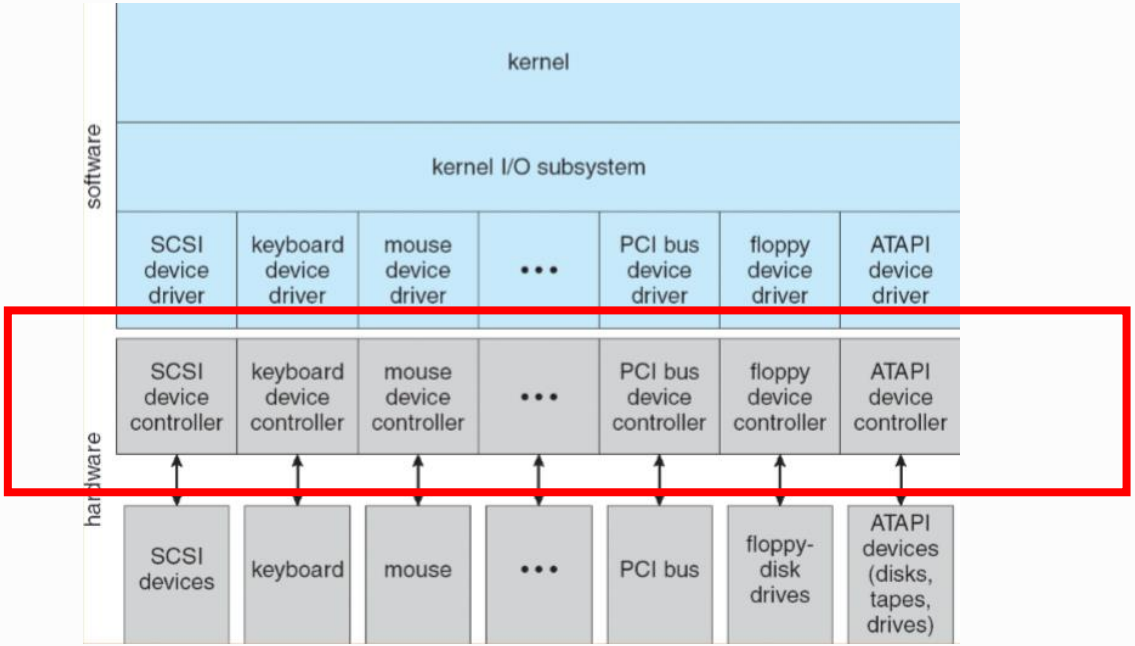
存储类

网络通信类

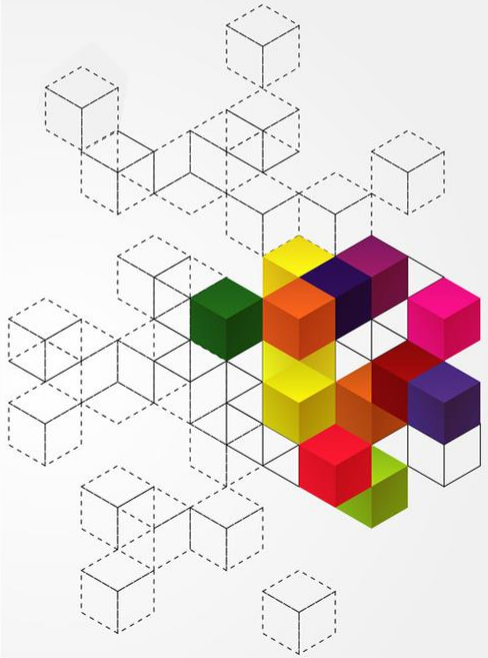


三、设备控制器

设备控制器是IO设备的控制中枢

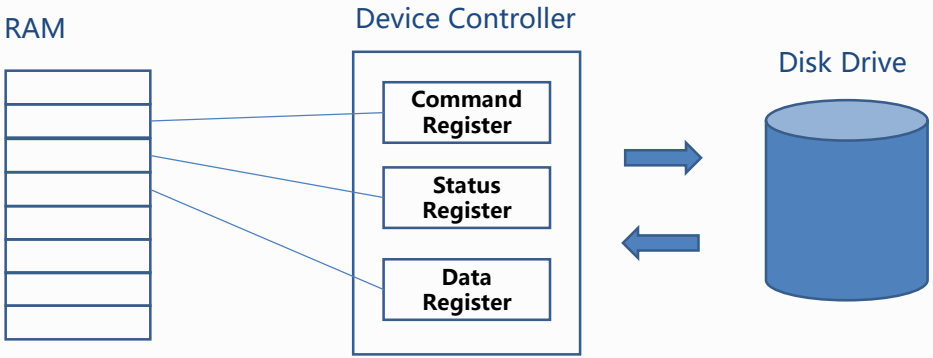


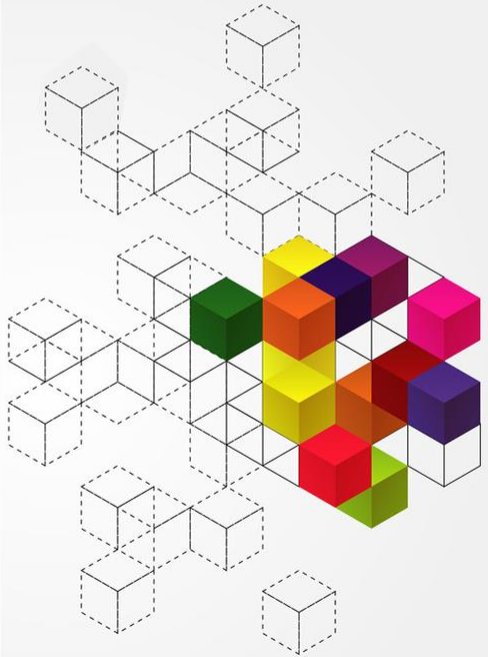
设备驱动程序，核心是对设备控制器进行编程



三、设备控制器

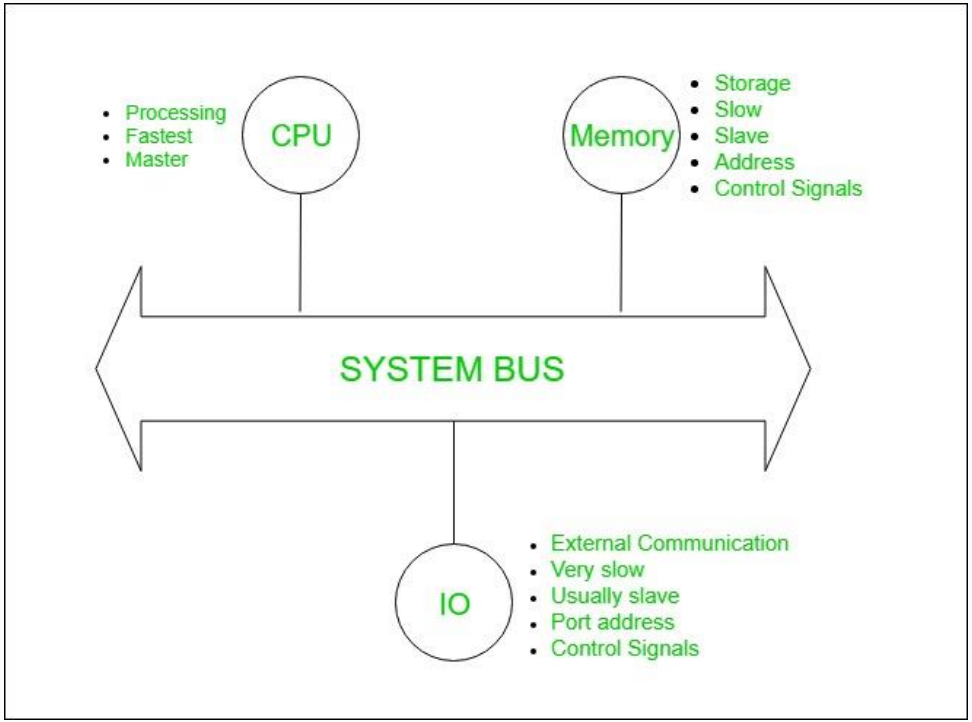
设备控制器示例：磁盘控制器

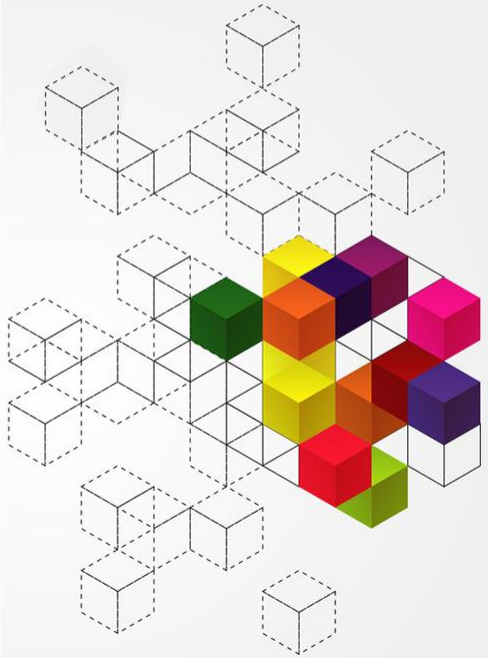




四、总线及其分类

系统总线（System Bus）

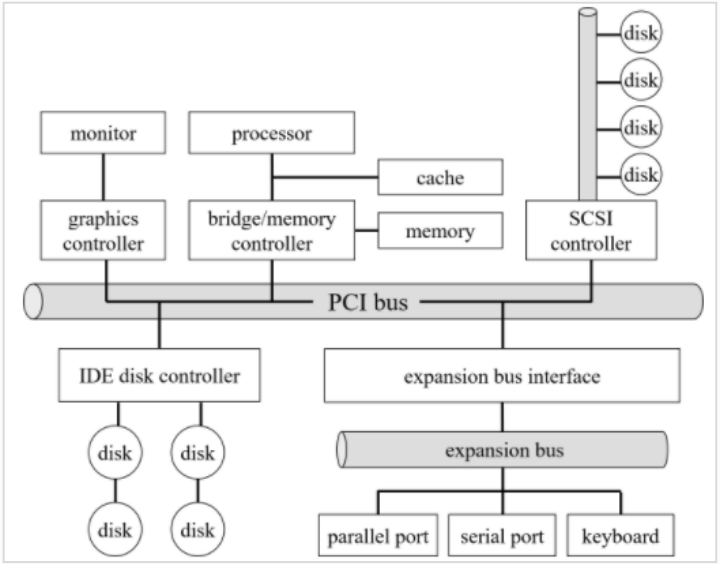


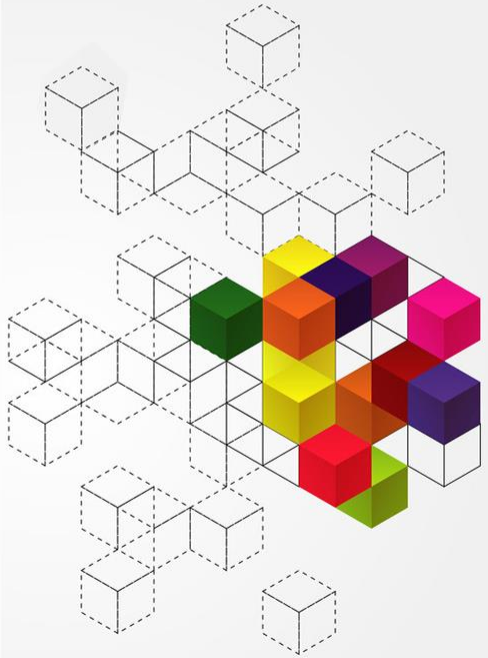


将CPU、memory、IO设备连接为一个整体

四、总线及其分类

典型的PC总线结构示意图





PCI: Peripheral Component Interface

四、总线及其分类

PC总线发展史

ISA,1982
4.77MB/s

EISA,1988
33.32MB/s

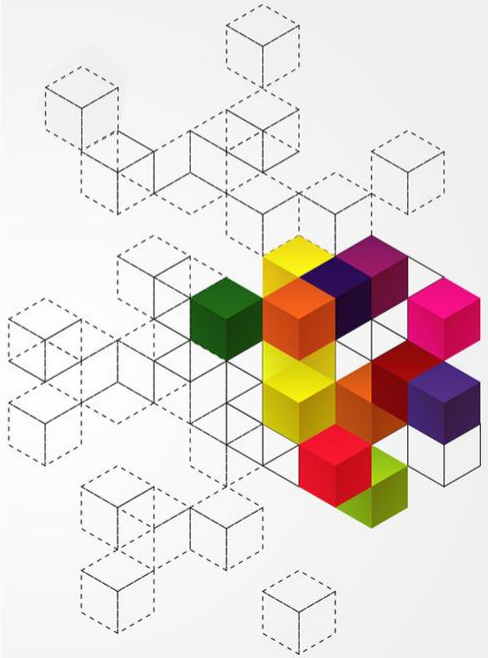
PCI,early 90s
133MB/s

PCIe 3.0, 2011
31.5GB/s

MCA,1987
40MB/s

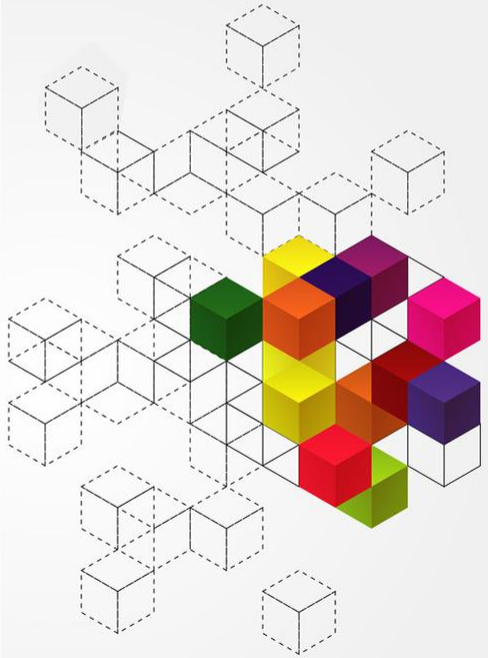
USB 2.0,2000
60MB/s

USB 3.0,2010
500MB/s



本讲小结

- IO子系统功能概述



内容纲要

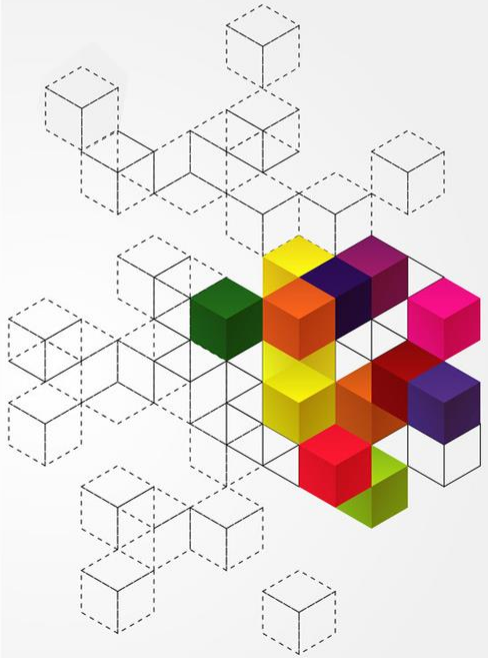
13.2 IO控制方式

一、 程序控制IO

二、 中断控制IO

三、 DMA控制

四、 通道方式



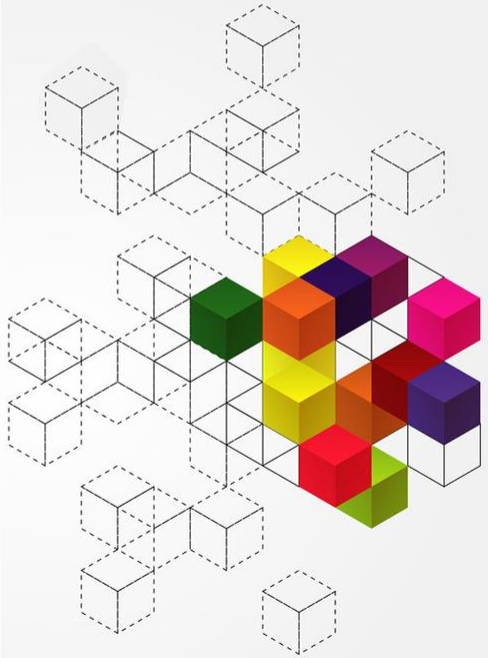
几种IO控制方式

通道控制IO

DMA

中断控制IO

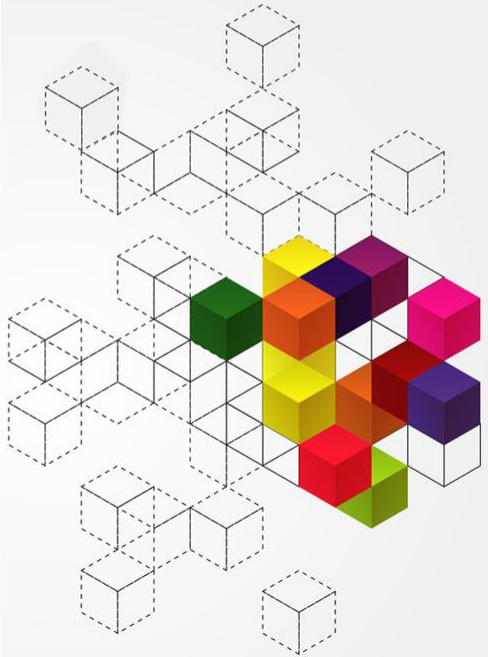
程序控制IO



一、程序控制IO

直接程序控制方式由用户进程直接控制主存或CPU和外围设备之间的信息传送。直接程序控制方式又称为轮询方式，或忙等方式。

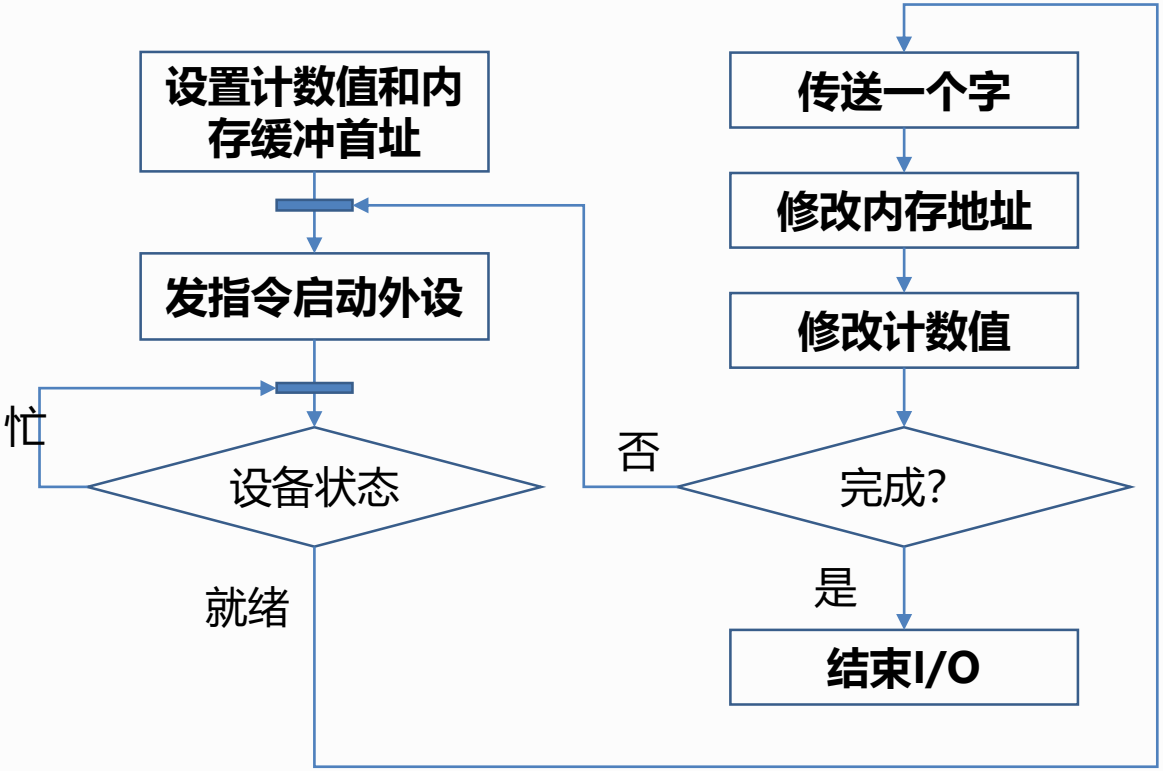
编程控制IO在旧接口的设备上使用
-串口
-非ECP模式的并口

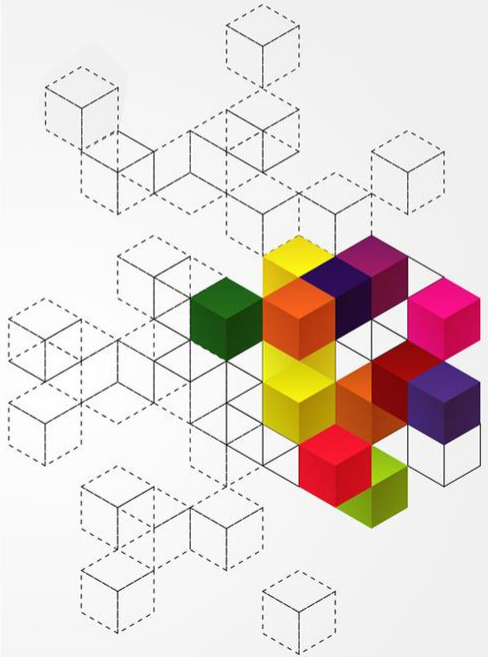


ECP（Extended Capabilities Port）

一、程序控制IO

程序控制IO: CPU指令控制整个IO过程

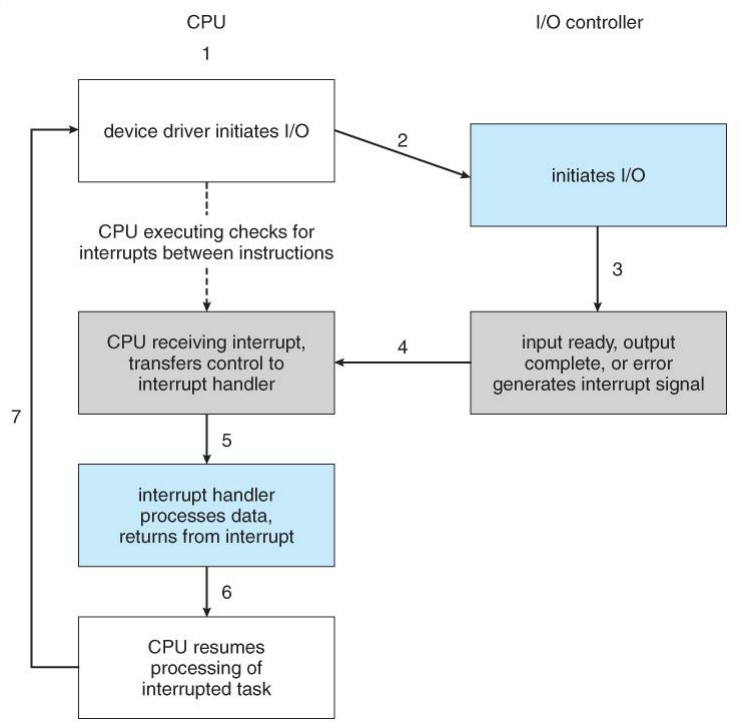


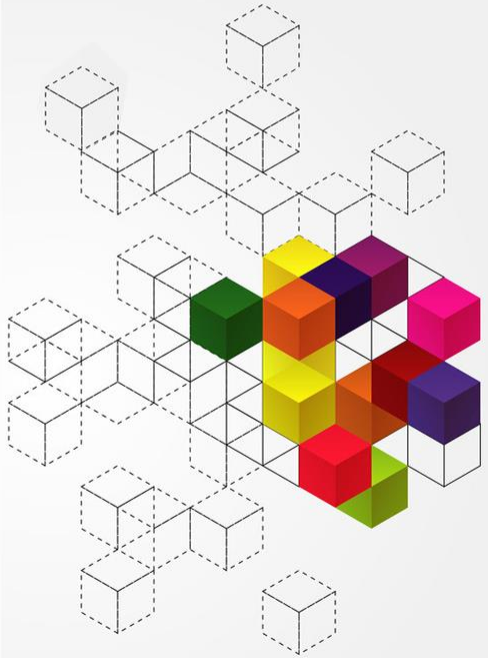


二、中断控制IO

Interrupt-driven IO
-需要中断硬件支持
-效率较编程控制IO高

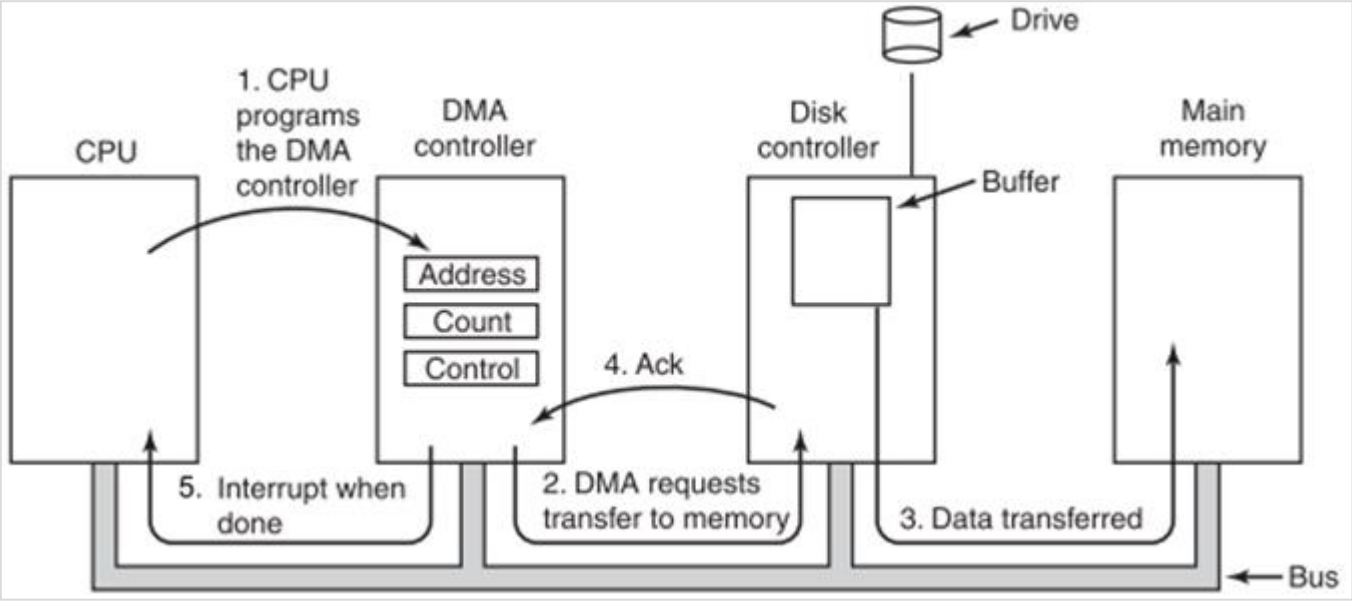
CPU与IO设备并发工作

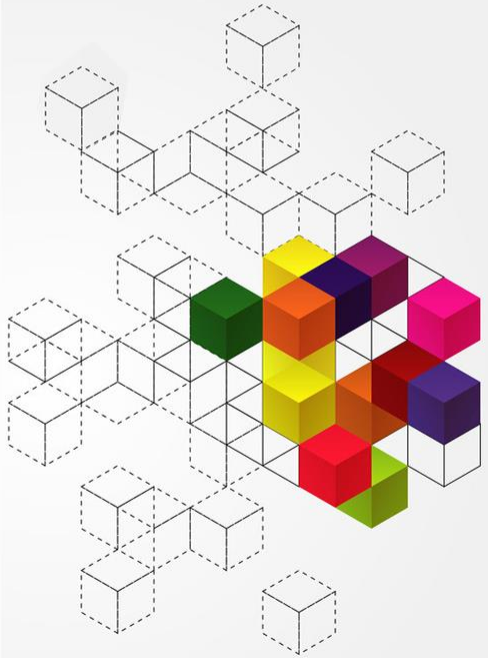




三、DMA控制

Direct Memory Access (直接内存访问)



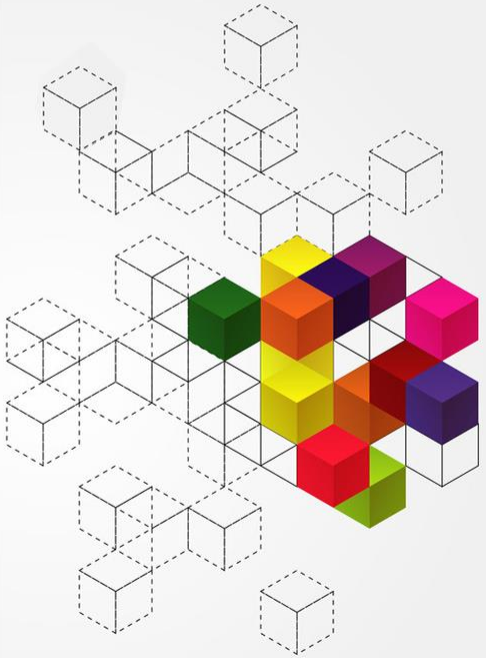


三、DMA控制

Cycle Stealing （周期窃取）

- DMA controller与CPU采取Interleaving(交替)使用memory resource

机器指令周期
1.IF：Instruction Fetch(must mem access)
2.DE：Decode
3.FO：Fetch Operand(optional mem access)
4.EX：Execute
5.WM：Write result to Memory(optional mem access)
●其中FO、WM这两个阶段，CPU可能不使用Memory与I/O Device之间的Data transfer
●万一CPU与DMA controller对memory存取发生conflict，则給DMA较高的优先权



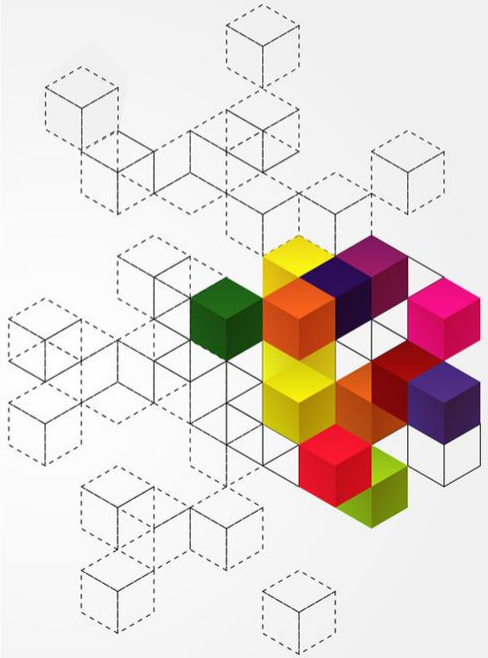
四、通道控制方式 – 背景

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

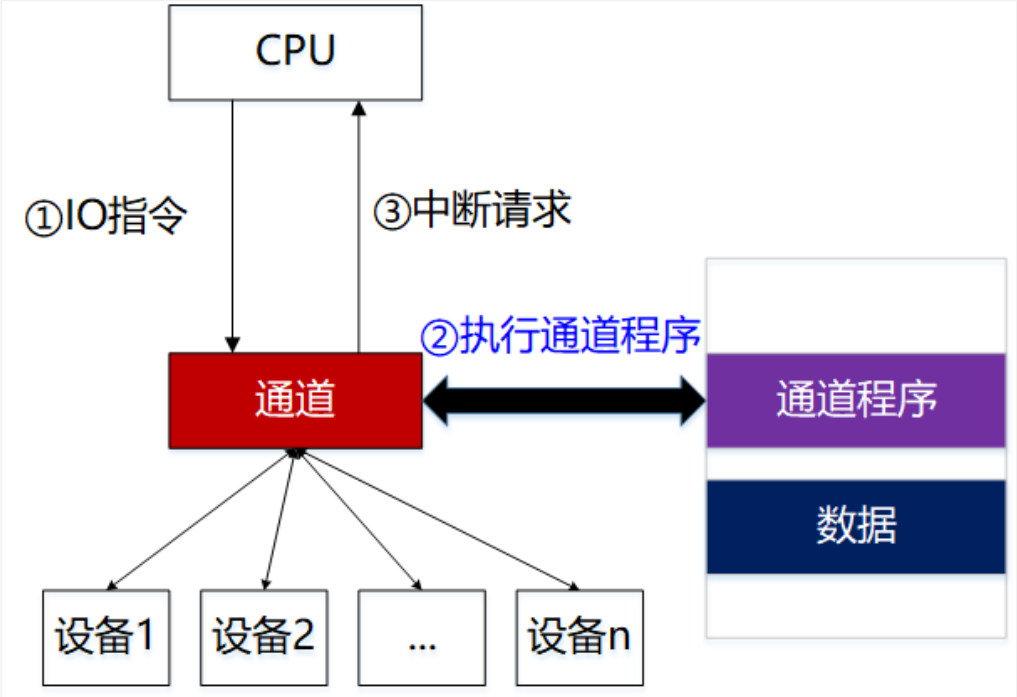
因此，大型计算机系统中采用通道处理机

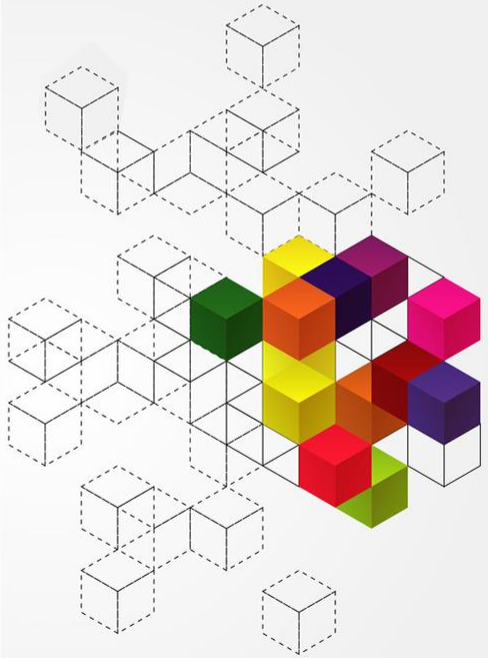
- 使CPU摆脱繁重的IO处理负担
- 共享IO接口



四、通道控制方式

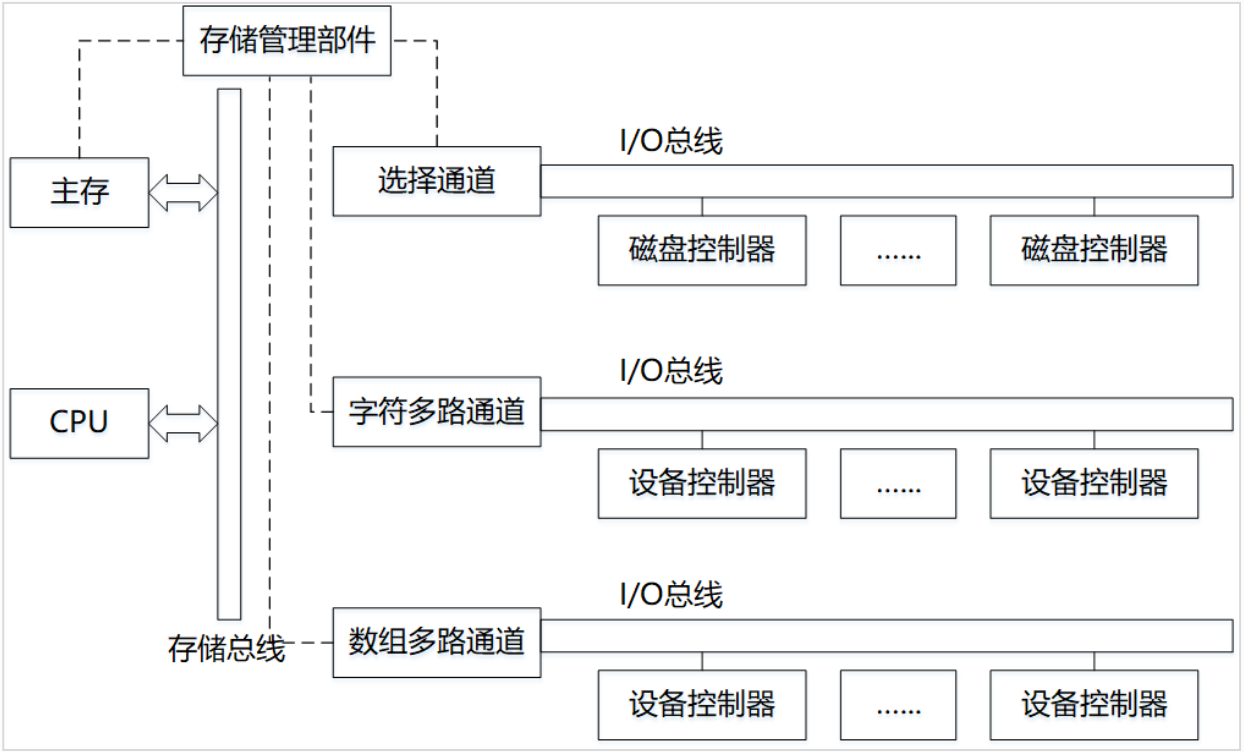
Channelled IO
通道控制

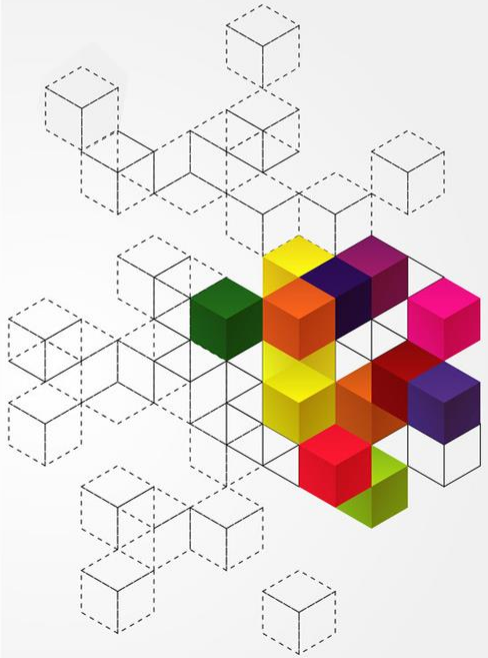




四、通道控制方式

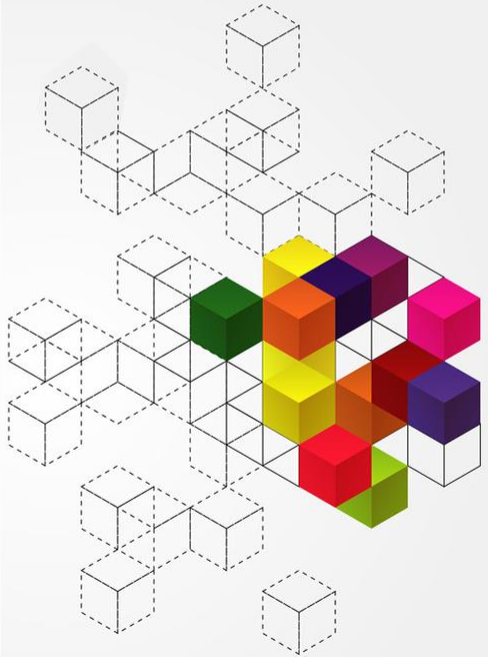
具有通道的计算机系统典型结构





本讲小结

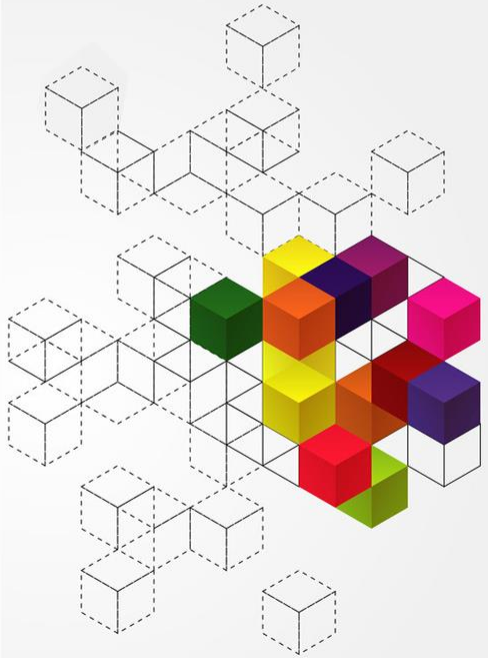
- 编程控制IO
- 中断控制IO
- DMA控制方式
- 通道控制方式



几种IO控制方式

下列IO控制方式中，哪一个基本不需要额外硬件支持?

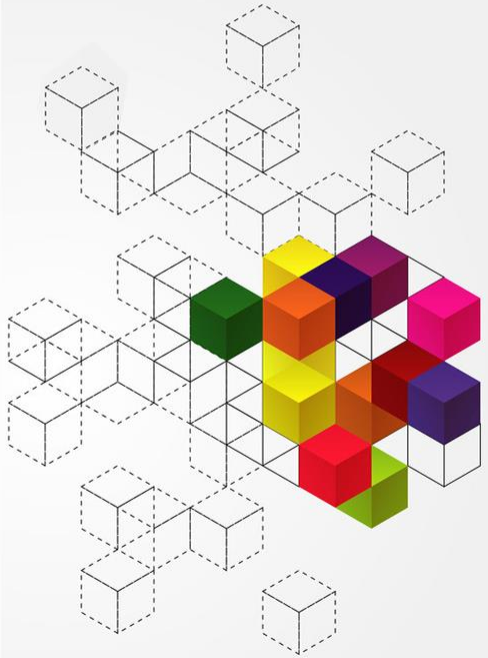
A.程序轮询方式
B.通道控制方式
C.中断控制方式
D.DMA控制方式



几种IO控制方式

下列选项中，（）不属于操作系统提供给普通用户的可使用资源?

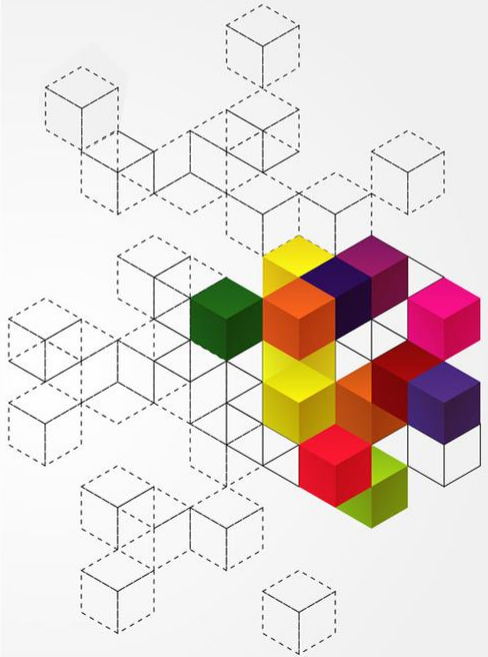
A.I/O设备
B.中断机制
C.存储器
D.处理器



几种IO控制方式

中断系统一般由相应的（ ）组成。

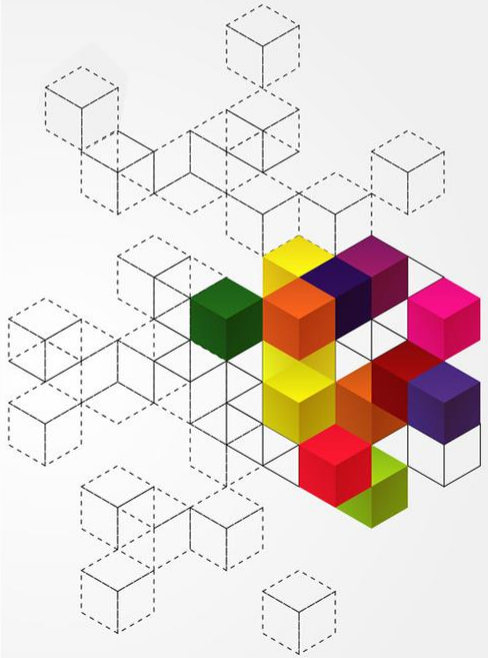
A.软件
B.固件
C.硬件和软件
D.硬件



几种IO控制方式

以下关于通道的叙述中，不正确的是（ ）。

A.通道是与DMA相同的一种I/O控制部件
B.通道能同时控制多台同类型或者不同类型的设备
C.通道方式中信息的传送是通过执行通道程序来完成
D.通道分为字符多路通道、数组多路通道和选择通道



内容纲要

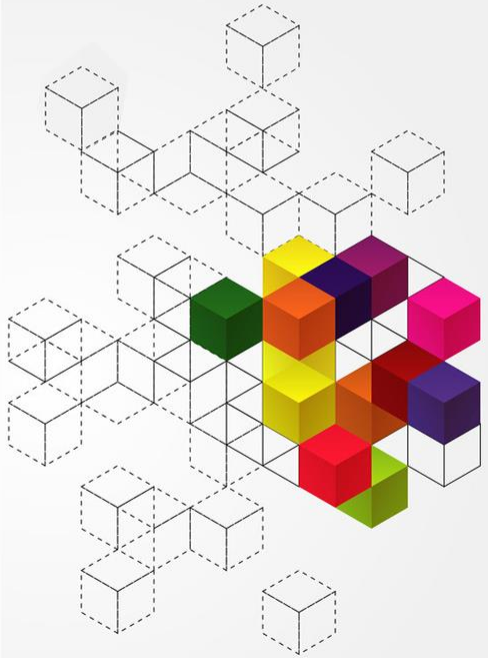
13.3 IO缓冲技术

一、IO子系统中的缓冲技术

二、单缓冲

三、双缓冲

四、环形缓冲

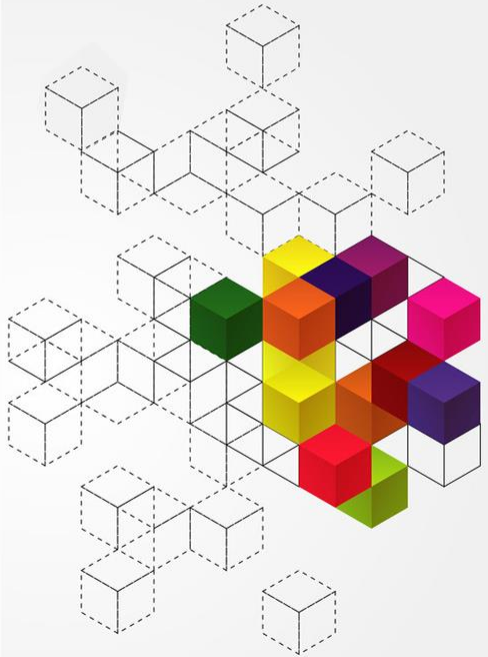


一、IO子系统中的缓冲技术

IO子系统中引入缓冲的目的

-应对CPU与外设间速度不匹配的矛盾

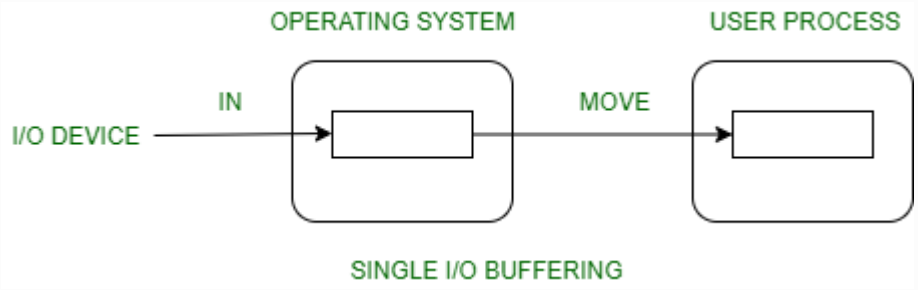
-解决逻辑记录与物理记录不匹配的矛盾

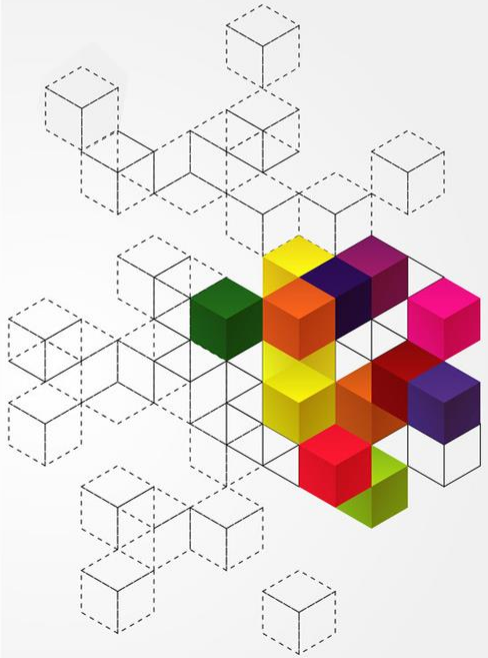


二、IO缓冲技术-单缓冲

Single Buffer

-提供足够大的单个缓冲区，可以减少访问设备的频次，提升效率

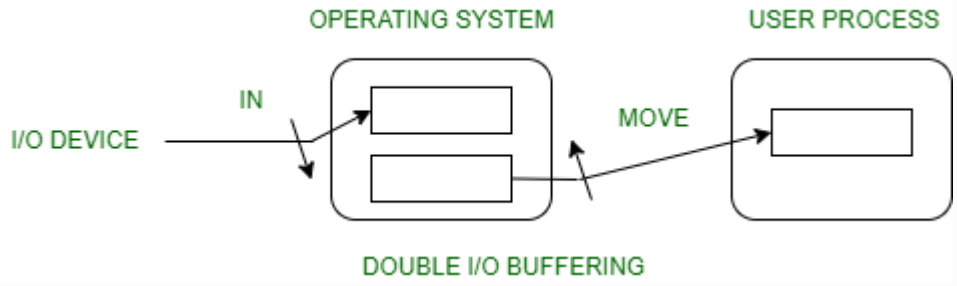


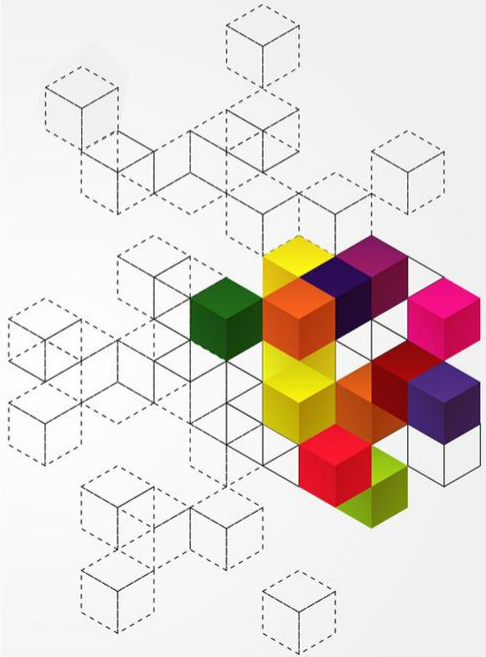


三、IO缓冲技术-双缓冲

Double Buffer

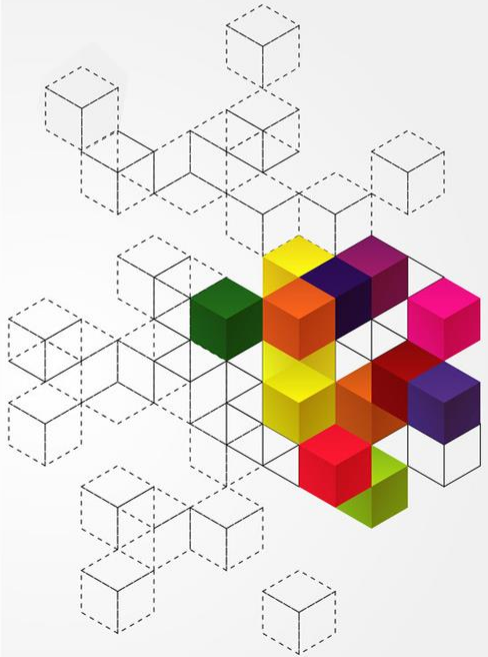
-对单缓冲的优化，通过增加一个系统buffer来获得数据输入与数据处理的并发





三、IO缓冲技术-双缓冲

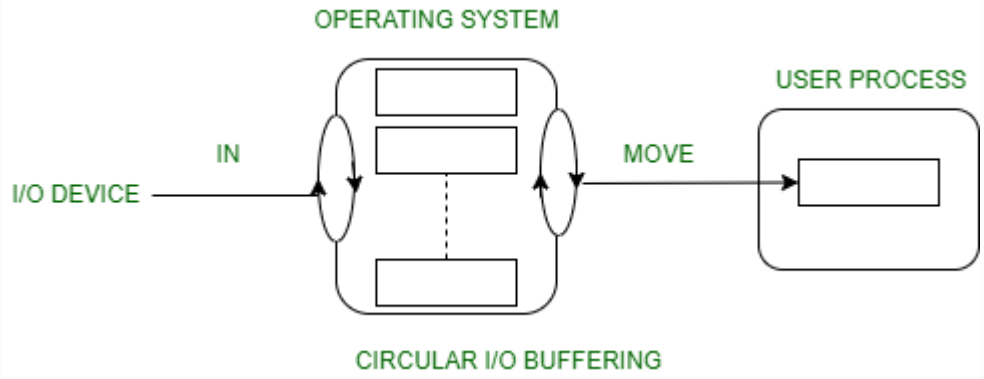
双缓冲依旧存在的不足：
-无法应对IO bursts （例如，网卡，可能突然有大量数据涌入，需要IO子系统处理)

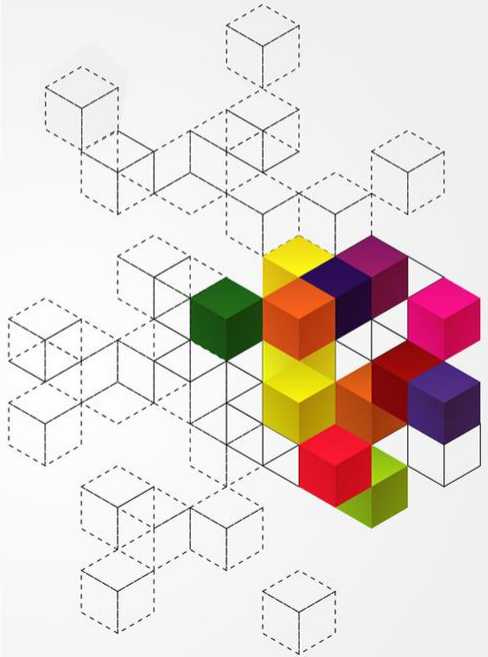


四、IO缓冲技术-环形缓冲

Circular Buffer

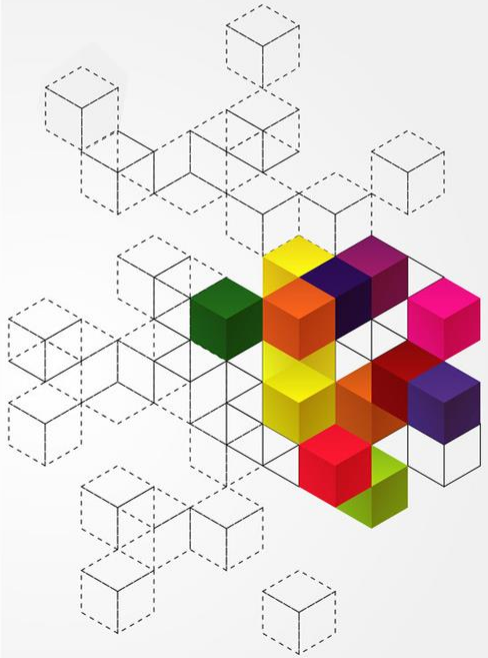
-有限环形缓冲，通过缓冲区数量的扩容，加上有限缓冲的并发处理，可以应对IO Bursts





本讲小结

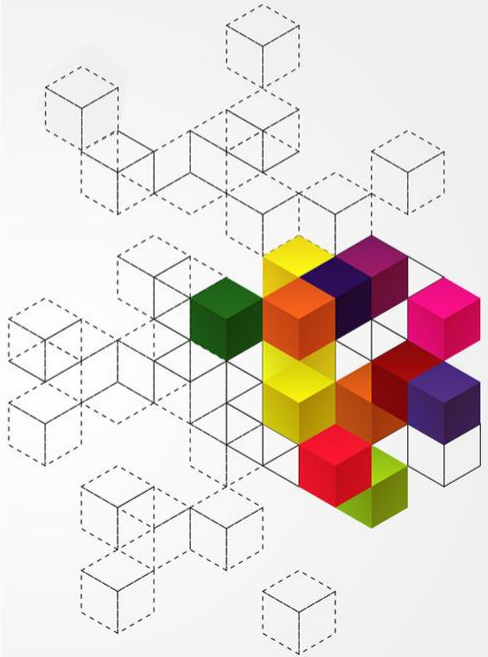
- IO缓冲技术概述
- 三种典型的缓冲模式
  - 单缓冲
  - 双缓冲
  - 环形缓冲



零、Linux的IO子系统中的缓存

- **Linux的IO子系统中，缓冲被重点进行设计，以提升IO子系统性能**

- **两类关键缓存：Page Cache和Buffer Cache**
  - Page Cache：以页为单位，缓存文件内容
  - Buffer Cache: 内核为了加速对底层存储介质的访问速度而构建的一层缓存



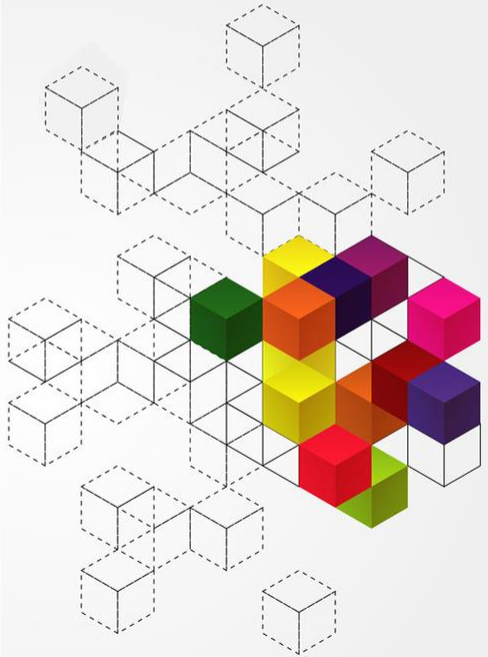
内容纲要

13.4 Linux IO缓冲机制

一、Buffer Cache

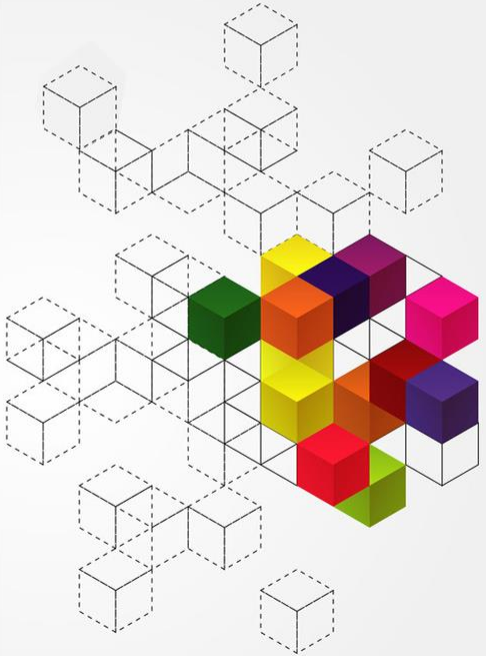
二、Page Cache

三、两类Cache演进历史



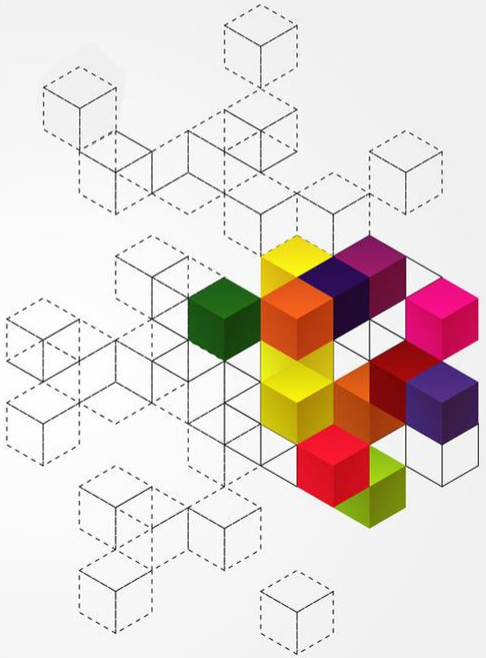
一、Buffer Cache

- **设立buffer Cache的目的**
  - 在内存中设立磁盘扇区数据的缓存
- **相关背景**
  - 磁盘的最小数据单位是扇区，每次读写磁盘都是以扇区为单位进行
  - 如果直接访问磁盘，那么意味着及时用户仅更新某个扇区一个字节的数据，他都必须更新整个扇区数据
- 提升效率的方法
  - 为磁盘扇区建立一层缓存，以扇区的整数倍大小构建缓存块



一、Buffer Cache

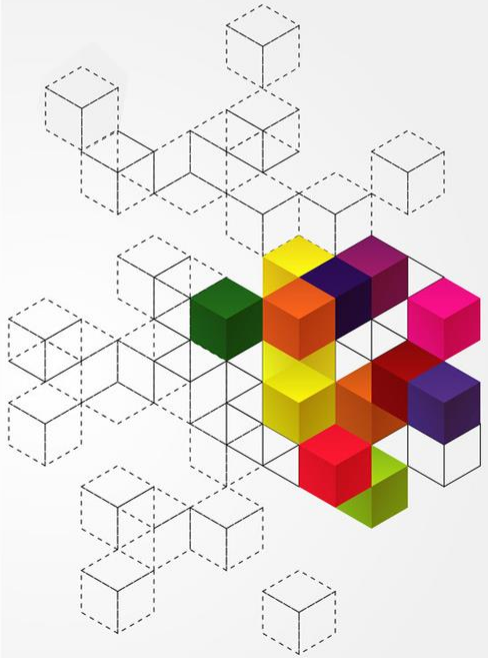
- **基于Buffer Cache的磁盘扇区数据读写**
  - 当首次访问某个扇区，在buffer cache中建立新的缓存项
  - 此后，对于该扇区的读写请求，直接从内存中读写
  - 通过异步方式，将更新后的数据写回对应磁盘扇区



二、Page Cache

- **Page Cache**
  - 缓存在Page Cache中的文件数据，能够更快地被读取
  - 进行写入操作时，数据可以在被写入Page Cache后立即返回

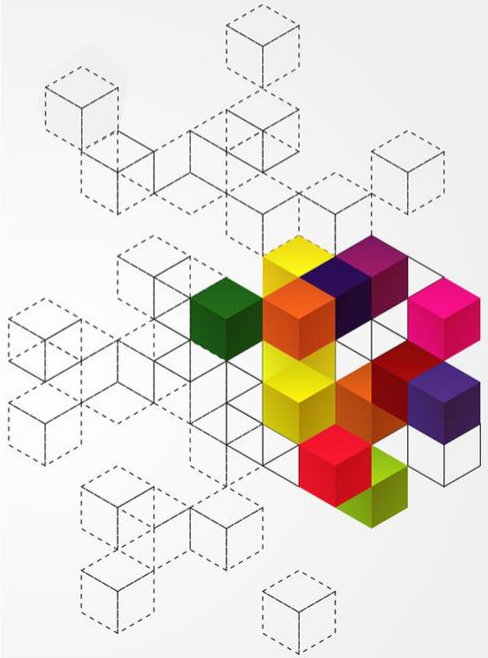
Page Cache可以大幅提高上层应用读写文件地整体性能



二、Page Cache

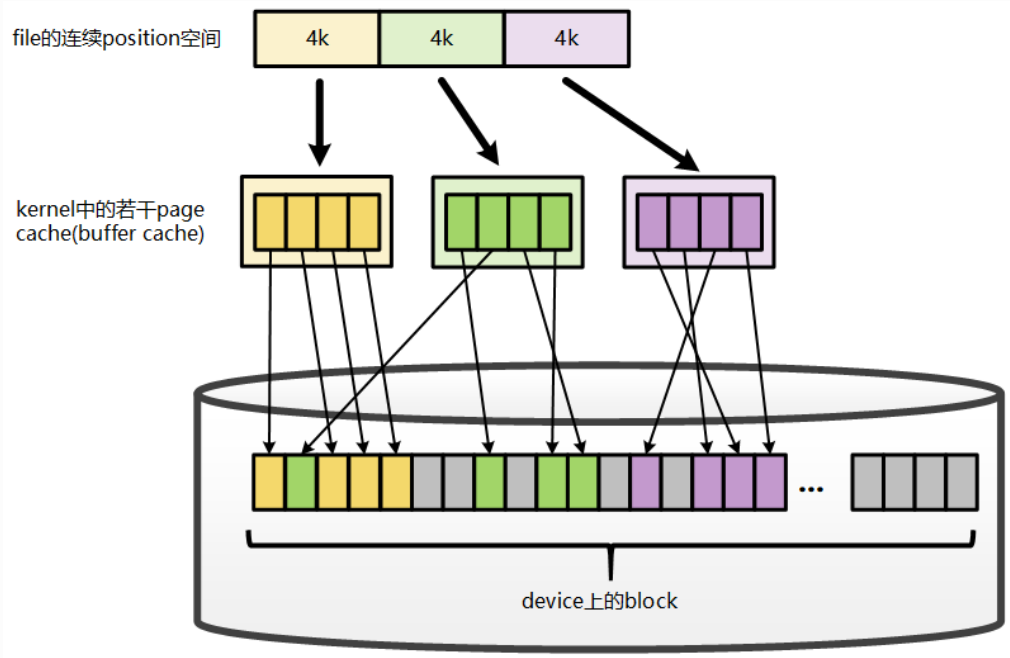
- **Page Cache与Buffer Cache的逻辑关系**
  - 参考代码：Linux 2.6.18

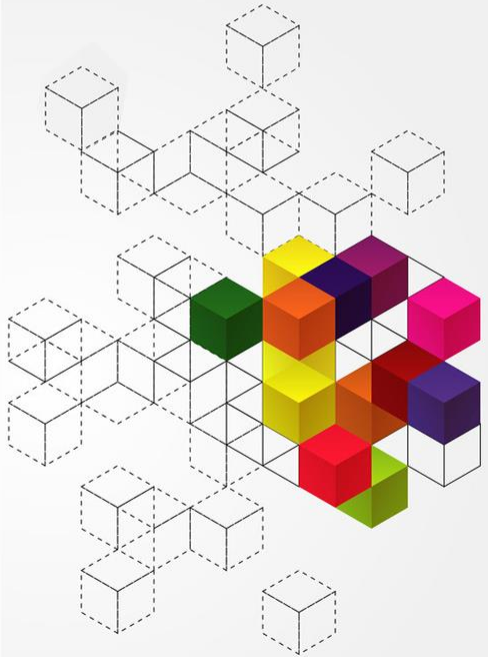
一个作为缓存的页，
对上层，它是某个File的一个Page Cache
对下，它是一个磁盘上的一组Buffer Cache



二、Page Cache

- **Page Cache与Buffer Cache的逻辑关系**
  - 参考代码：Linux 2.6.18





三、两类Cache的演进历史

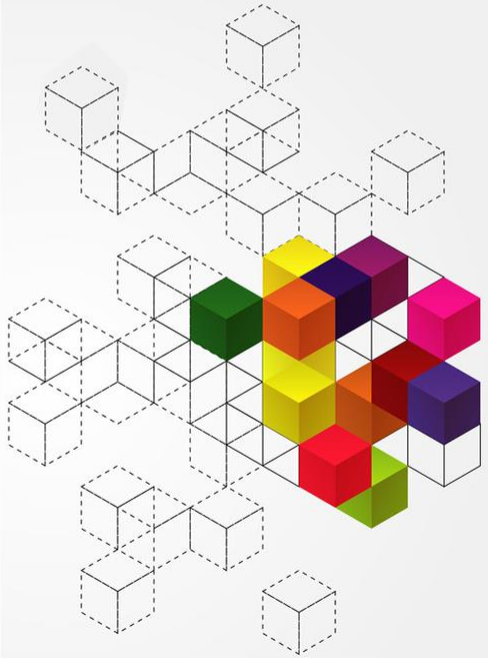
- **第1阶段：仅有Buffer Cache**
  - 参考Linux版本：Linux 0.11

在Linux-0.11的代码中，buffer cache是完全独立的实现

●其中还没有基于page划分内存单元，而是以原始指针的系形式出现

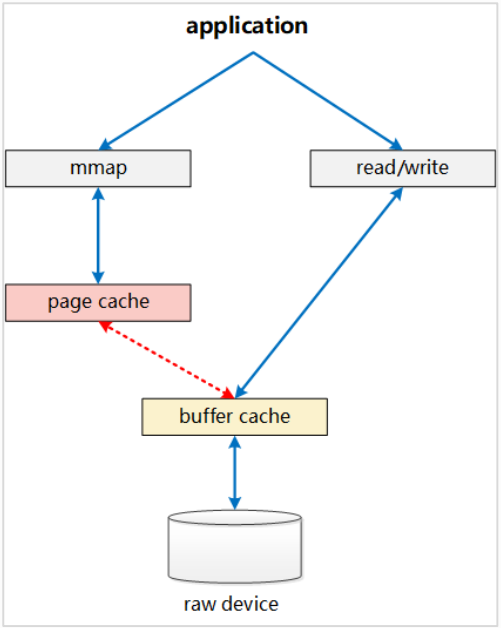
●每一个block sector，在kernel内部对应一个独立的buffer cache单元，这个buffer cache单元通过buffer head来描述

```
68: struct buffer_head {
69:     char * b_data;          /* pointer to data block (1024 bytes) */
70:     unsigned long b_blocknr;    /* block number */
71:     unsigned short b_dev;       /* device (0 = free) */
72:     unsigned char b_uptodate;
73:     unsigned char b_dirt;       /* 0-clean,1-dirty */
74:     unsigned char b_count;      /* users using this block */
75:     unsigned char b_lock;       /* 0 - ok, 1 -locked */
76:     struct task_struct * b_wait;
77:     struct buffer_head * b_prev;
78:     struct buffer_head * b_next;
79:     struct buffer_head * b_prev_free;
80:     struct buffer_head * b_next_free;
81: };
82:
```

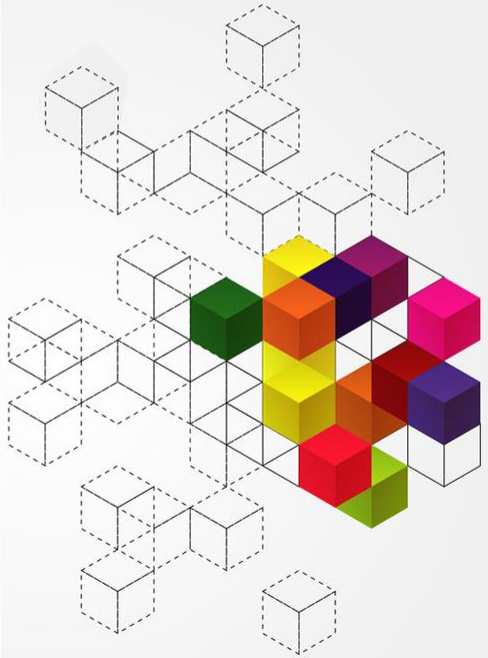


三、两类Cache的演进历史

- **第2阶段：Buffer Cache与Page Cache并存**
  - 参考Linux版本：Linux 2.2

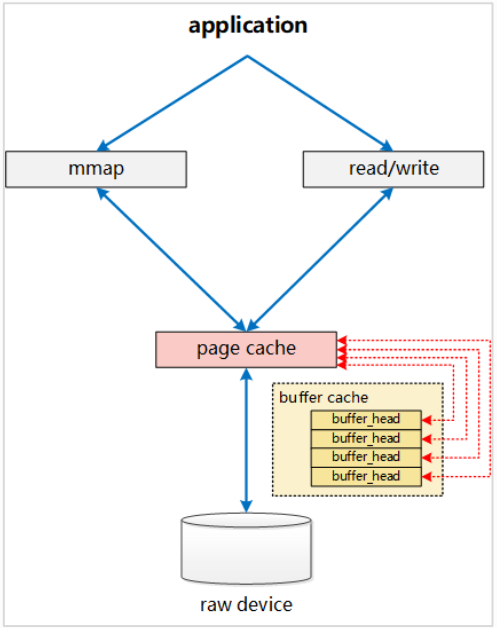


Linux-2.2中，磁盘读写操作访问的高速缓冲仍然是Buffer Cache。其访问模式与上面Linux-0.11版本的访问逻辑基本类似
但此时，Buffer Cache已基于page来分配内存，buffer_head内部，已经有了关于所在page的一些信息



三、两类Cache的演进历史

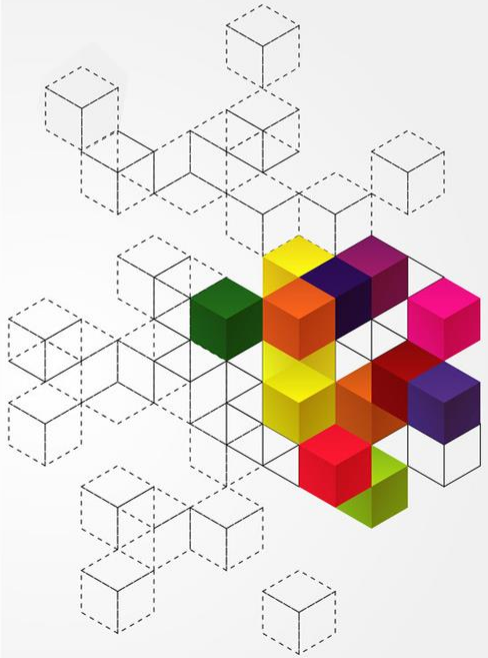
- **第3阶段：Buffer Cache与Page Cache融为一体**
  - 参考Linux版本：Linux 2.4.0



Linux-2.4版本中对Page Cache、Buffer Cache的实现进行了融合

●融合后的Buffer Cache不再以独立的形式存在，Buffer Cache的内容，直接存在于Page Cache中

●保留了对Buffer Cache的描述符单元：buffer_head



本讲小结

Unix/Linux IO缓冲机制

- ✓Buffer Cache
- ✓Page Cache
- ✓两类Cache的演进历史