操作系统原理复习提纲

一、选择/名词/填空(工程知识/概念部分)：举例：“临界区”是指：

A. 一组临界资源的集合　　B. 可共享的一块内存区

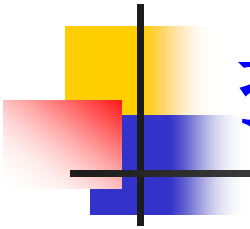
C. 访问临界资源的一段代码　　D. 请求访问临界资源的代码

二、简答题（工程知识/原理部分）

举例：简述文件管理系统中引入OPEN操作和CLOSE操作的目的及其实现的主要步骤？

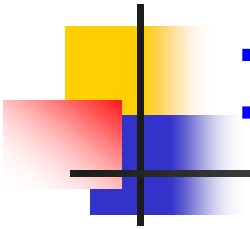
三、计算/解答题（问题分析与解决/设计能力）

举例：现有四个进程R1,R2,W1,W2，它们共享一个缓冲区（只能放一个数）。进程R1每次把从键盘上读入的一个数存到该缓冲区中，供进程W1打印输出；进程R2每次从磁盘上读一个数存放到该缓冲区中，供W2打印输出。当一个进程把数存放到缓冲区后，在该数还没有被打印输出之前不准任何进程再向缓冲区中存数。当一个进程已把缓冲区中的数打印输出后，在缓冲区中还没有存入一个新的数之前不准任何进程再从缓冲区中取数打印。用ＰＶ操作来协调它们的工作。



操作系统原理复习提纲

- **Introduction**
- **Operating System Structures**
- **CPU管理**
- 存储器管理
- 文件管理
- 设备管理



Introduction

- 操作系统的定义
- 操作系统的目标和作用
- 操作系统的功能
- 操作系统的分类
- 操作系统的特征

操作系统的定义

- 操作系统是一组控制和管理计算机系统中的各种软硬件资源，合理地组织计算机系统的工作流程，方便用户使用的程序的集合。

操作系统的目标和作用

- **convenience方便性:**方便用户使用计算机系统;
- **efficiency有效性:**提高系统资源的利用率;
- 可扩充性、开放性;

- 资源管理观点----OS是系统资源管理者，用于控制和管理计算机系统的硬件和软件资源。
- 用户观点----OS是用户与裸机之间接口

操作系统功能

- 从资源管理的角度
  - 处理机管理
  - 存储器管理
  - 设备管理
  - 文件管理
  - 用户接口

操作系统的分类

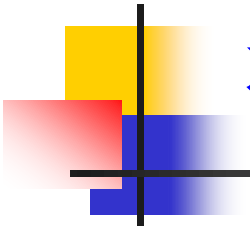
- 批处理操作系统：单道、多道
- 分时操作系统
- 实时操作系统
- 通用操作系统

操作系统的特征

- 并发：与并行的区别
- 共享：资源互斥使用（临界资源）、同时使用
- 虚拟：一个物理实体映射为多个逻辑实体。
- 异步性：在多道程序环境下，每道程序的推进时间与顺序受运行环境的影响，是不确定的；程序执行结果不确定

相关技术

- 多道程序设计：在内存中同时存放多道用户作业，使它们都处于执行的开始点和结束点之间。
  - 特点：多道、宏观上并行、微观上串行
- 分时技术：把处理机的运行时间分成很短的时间片，按时间片轮流把处理机分配给各作业使用。



引论复习题

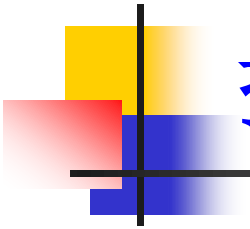
- 基本概念
  - 分时系统
  - 实时系统
  - 多道程序设计

引论复习题

- 填空
  - 操作系统的五大功能是__、、、、、__。
  - 如果一个**OS**兼有批处理、分时处理、实时处理**OS**三者中的两者，这样的**OS**称为_____。
  - 多道程序设计是利用了___和 ___的并行工作能力来提高系统效率的。
  - 如果**OS**具有很强的交互性，它可供多个用户使用，但时间响应不太及时，则属于___类型；如果**OS**可靠，时间响应及时但仅有简单的交互能力，则属于类型；如果**OS**在用户提交作业后不提供交互能力，只提供作业流程的自动化，则属于___类型。
  - 实时系统应具有两个基本特征：_____和_____。

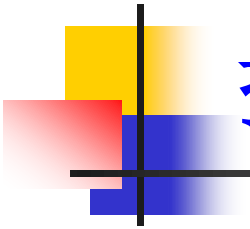
引论复习题

- 简答题：
  - **OS**为实现并发、共享的特性，必须解决哪些问题？
  - 为保证多道程序的正确运行，在技术上要解决哪些基本问题？
  - 简述批处理系统、分时系统、实时系统各自的特点。
  - **OS**中采用多道程序设计技术，带来什么好处**?**



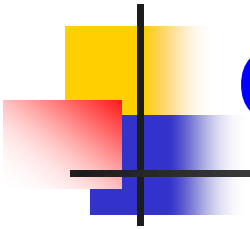
操作系统接口

- 作业级接口
  - 命令行
  - **GUI**
  - 批处理
- 程序级接口
  - 系统调用



操作系统结构

- 简单结构:**DOS, 早期UNIX**
- 层次结构:**THE**
- 微内核结构:**MACH**



CPU管理

- 进程的概念
- 进程的控制
- **thread线程**
- 进程的互斥与同步
- **CPU调度**
- **deadlock死锁**

一、进程的概念

1，引入进程的目的：

- 在多道程序设计的环境下，程序是并发执行的，它破坏了程序的封闭性和可再现性，使得程序和计算不再一一对应
- 且由于资源共享，导致在各个程序之间可能存在相互制约的关系，出现了许多新的特征：动态性、并发性、独立性和异步性。
- 程序(进程)并发的特点:间断性\失去了封闭性\不可再现性
- 程序这个静态概念已经不能如实反映程序活动的这些特征。为此引入进程这个概念来描述系统和用户的活动。

一、进程的概念

2. 进程概念：

- 进程是程序的一次执行过程，是系统进行资源分配和调度的一个基本单位。

3. 进程的特征

- 动态性：有生命周期
- 并发性：并发执行
- 独立性：独立获得资源、独立运行单位
- 异步性：推进速度不可预知道、执行结果不确定
- 结构性：由程序段、数据段和**PCB**组成

一、进程的概念

4，进程与程序、进程与线程的区别

- 进程与程序

（1）进程是程序的一次执行，是一个动态的概念；而程序是一组有序的指令， 是一种静态的概念。即进程是程序执行的动态过程，而程序则是进程运行的静态文本。

（2）一个进程可以执行一个或几个程序，同一个程序也可能由几个进程同时执行。

（3）程序可长期保存，而进程是有生命周期的。

（4）进程是并发实体，而程序则不是。

一、进程的概念

进程与线程：

（1）调度方面：线程作为调度分派的基本单位，而进程则是资源分配和调度的一个基本单位；

（2）并发性方面：进程之间可以并发，一个进程的多线程之间也可并发执行；

（3）拥有资源方面：进程作为拥有资源的基本单位，而线程只拥有少量必不可少的资源，但它可以访问所属进程的资源

（4）系统开销方面：进程切换要涉及到进程环境的切换，开销较大，而线程间切换只需保存和设置少量的寄存器内容，开销远小于进程切换开销。

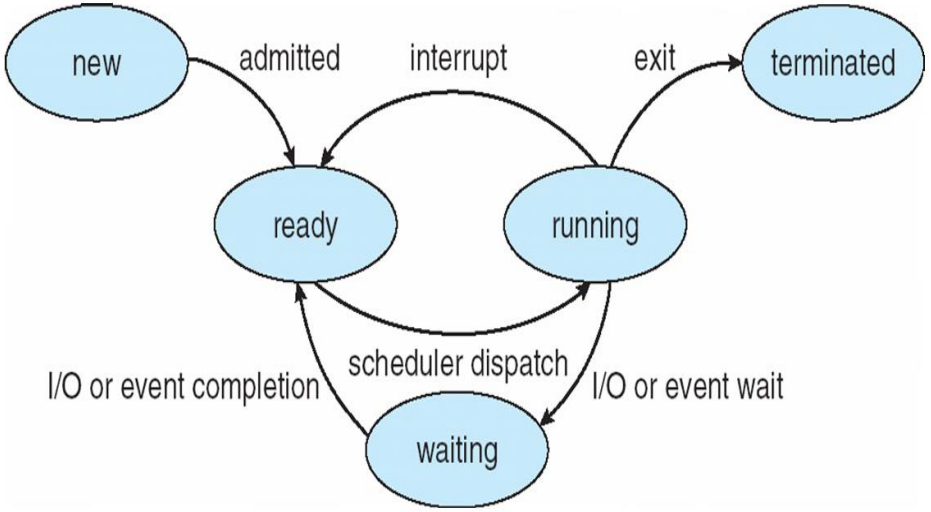
一、进程的概念

5，进程的组成：程序、数据、PCB

- **PCB**：记录**OS**所需的、用于描述进程的当前情况以及控制进程运行的全部信息，是进程存在的唯一标志，常驻内存。
- **PCB**的内容：
  - 进程标识符
  - 处理机状态（**CPU**现场）
  - 进程调度信息：状态、优先级、时间、事件
  - 进程控制信息：地址、通信信息、资源

一、进程的概念

6.进程的状态与控制



- 进程控制是由OS内核完成;
- OS内核通过执行相应的原语来实现进程的控制;
- 控制原语：创建、终止、阻塞与唤醒、挂起与激活；

三、进程的互斥与同步

- 进程间关系：间接（互斥）、直接（同步）
- 临界资源、临界区
- 进程的互斥：进程在运行中争用系统资源，对于独占型资源，只能一个进程使用完，另一个进程才能使用。
- 进程的同步：在异步环境下，互相合作的进程按各自独立的速度向前推进，但在某些确定点上必须协调工作。即当某个进程到达这些点后，等待另一进程发来信息，否则就只能停下来等待其操作的完成，进程间的这种协同关系称为进程同步。

三、进程的互斥与同步

进程互斥的实现：

- ✓ **Peterson's Solution**
- ✓ **Synchronization Hardware**

进程同步与互斥的实现：

- 信号量与**P**、**V**操作
  - 信号量:只能执行初始化和**P**、**V**操作原语
  - **P**操作(**wait**)：**request a resource**
  - **V**操作(**signal**)：**release a resource**
- 记录型信号量的**P**、**V**操作定义
- 管程;

三、进程的互斥与同步

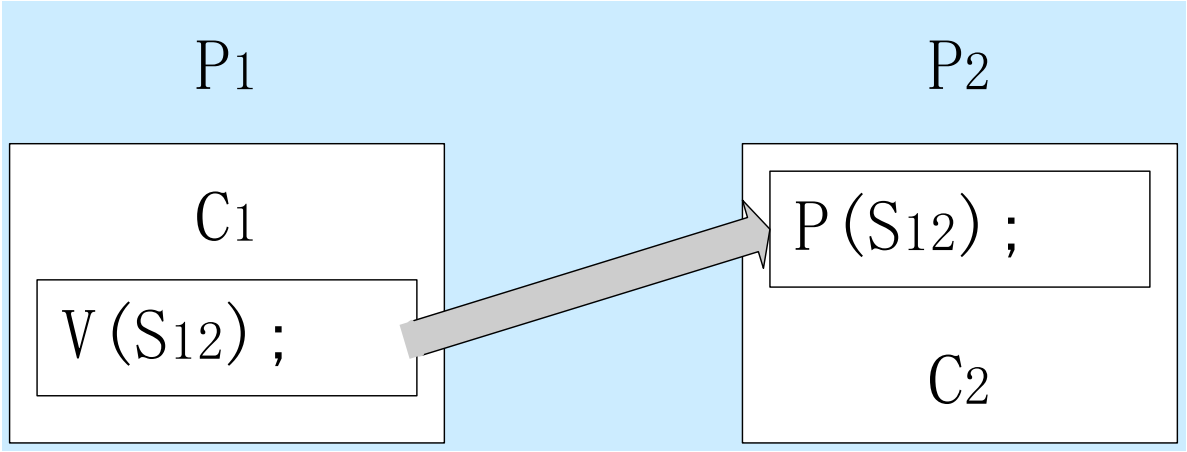
- 信号量的应用
  - 实现互斥;
  - 实现同步
- 经典IPC问题;
  - **Producer-consumer**
  - **Reader-writer**
  - **Dining-philosopher**

利用信号量实现互斥

```
P(mutex);
  critical section
V(mutex);
  remainder section
```

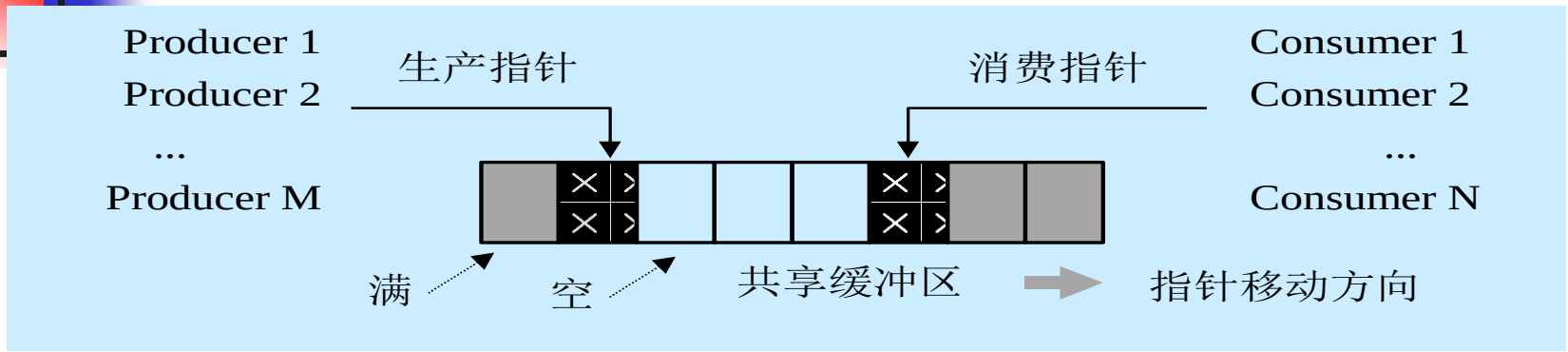
- 为临界资源设置一个互斥信号量**mutex(MUTual Exclusion)**，其初值为**1**；在每个进程中将临界区代码置于**P(mutex)**和**V(mutex)**原语之间;

利用信号量来描述前趋关系(同步)



- 前趋关系：并发执行的进程**P1**和**P2**中，分别有代码**C1**和**C2**，要求**C1**在**C2**开始前完成；
- 为每个前趋关系设置一个信号量**S12**，其初值为**0**

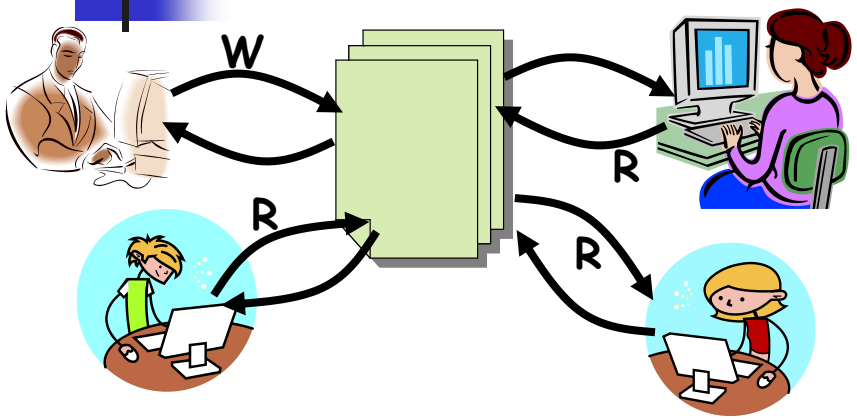
the producer-consumer problem



- 生产者和消费者分别设信号量：empty是“空”数目，初值为N，full是"满"数目，初值为0，实际上，full + empty == N
- mutex用于访问缓冲区时的互斥，初值是1；
- 多个P操作的顺序重要：先检查资源数目，再检查是否互斥——否则可能死锁

Producer	Consumer
P(empty); P(mutex); //进入区	P(full); P(mutex); //进入区
one unit --> buffer;	one unit <-- buffer;
V(mutex); V(full); //退出区	V(mutex); V(empty); //退出区

the readers-writers problem



- Wmutex表示"允许写"，初值是1。
- Rcount表示“正在读”的进程数，初值是0；
- Rmutex表示对Rcount的互斥操作，初值是1。

- “读－写”互斥，
- “写－写”互斥，
- “读－读”允许

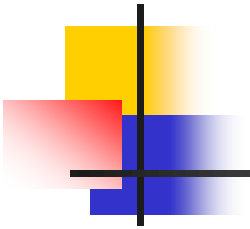
Writer

```
P(Wmutex);
  write;
V(Wmutex);
```

Reader

```
P(Rmutex);
    if (Rcount == 0)
      P(Wmutex);
  ++Rcount;
V(Rmutex);
  read;
P(Rmutex);
  --Rcount;
    if (Rcount == 0)
      V(Wmutex);
V(Rmutex);
```

读者优先



第一类：读者优先

- 如果读者来
  - 无读者、写者，新读者可以读；
  - 有写者等，但有其它读者正在读，则新读者也可以读；
  - 有写者写，新读者等；
- 如果写者来
  - 无读者，新写者可以写；
  - 有读者，新写者等待；
  - 有其它写者，新写者等待；

四、线程thread

- 引入线程目的：提高系统效率、提高系统资源利用率、减少进程并发执行时所付出的时空开销，使**OS**具有更好的并发性。
- 线程概念：又称轻型进程，进程内一个可调度的实体，处理机调度的基本单位。
- **Three multi-threading models:**
  - **Many-to-One**
  - **One-to-One**
  - **Many-to-Many**
- 用户级线程和内核支持线程：
  - 用户线程：存在于用户空间中，其创建、撤消和切换都不需系统支持。
  - 内核支持线程：是依赖于内核的，其创建、撤消和切换都是由内核实现的。

五、CPU Schedule

- 三级调度：
  - 高级调度：作业调度(Long-term Scheduling)
  - 低级调度：进程调度(Short-term Scheduling)
  - 中级调度：交换调度(Medium-term Scheduling)
- 调度方式：Preemptive、Nonpreemptive
- 调度时机
  - 现运行进程任务完成或出现异常
  - 现运行进程因某种原因由执行变成阻塞状态
  - 时间片用完
  - 采用可剥夺调度方式时，有更高优先级进程进入就绪队列

五、CPU Schedule

- **FCFS: The simplest;**
- **SJF: providing the shortest average waiting time;**
- 基于优先权的调度算法;
- 时间片轮转法: **more appropriate for time-shared system;**
- 多级队列法: **allows different algorithms to be used for various classes of processes;**
- **Real-time Scheduling：Rate-Monotonic Scheduling（ RMS ）/Earliest-Deadline-First (EDF) Scheduling;**
  - 算法需考虑的指标:
    - 周转时间**turnaround time**
    - 等待时间**waiting time**
    - 响应时间**response time**

六、deadlock

1.死锁概念

2. 产生死锁的原因**:**

- 系统资源不足；
- 进程推进顺序不合适；

3. 产生死锁的必要条件：

- 互斥使用**Mutual exclusion**；
- 不可剥夺**No preemption**；
- 请求保持**Hold and wait**；
- 环路等待**Circular wait**；

六、deadlock

4. 解决死锁的策略：

- 设计无死锁的系统：静态预防、动态避免；
- 允许出现死锁然后排除：检测并解除

5. 死锁的预防：破坏四个必要条件之一，通常是破坏第三、四个条件。

（1）资源的静态分配方法：进程运行前一次性申请全部资源,破坏请求和保持条件.

（2）资源的顺序分配法：系统的全部资源进行编号，只允许按编号顺序递增地申请，破除环路待。

（3）一个已占有资源的进程,若要申请新的资源，必须先放弃已占有的资源，破坏请求保持条件。

六、deadlock

6. 死锁的避免：当进程申请资源时，根据一定算法判断系统是否处在安全状态

- 安全状态：当多个进程动态申请资源时，系统按某一顺序逐次地为每个进程分配所需资源，使每个进程都可以在最终得到最大需求量后依次顺利完成；否则为不安全状态；让系统在动态分配资源的过程中，不要进入不安全状态。
  - 银行家算法：避免死锁的算法
    - 最大需求矩阵**MAX**
    - 已分配资源矩阵**ALLOCATION**
    - 尚需申请的资源矩阵**NEED**
    - 请求向量**Request[i]**
    - 目前可用资源**Available[j]**
    - 工作向量**Work**

六、deadlock

7. 死锁的检测

- 确定死锁是否存在，检测出死锁，并想办法解除

8. 死锁的解除：

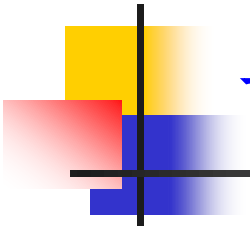
- 删除法：删除死锁进程，将其资源分给其他进程
- 剥夺法：剥夺某些进程的资源

CPU管理复习要点

- 进程概念、基本状态及转换
- 进程同步与互斥
- 线程概念、分类、与进程比较
- 处理机三级调度概念、调度算法
- 用信号量实现进程间的同步与互斥
- 管程概念、实现
- 死锁概念、原因、必要条件、解决方法

存储器管理

- 重定位：静态、动态，如何实现；
- 动态分区分配方式：分配算法（空闲分区数据结构的组织、分配和回收）、分区保护；
- 分页和分段存储管理：原理、地址变换机构、页（段）表、快表、缺页（段）中断的处理、页面置换算法、信息共享和保护；
- 虚拟存储器：引入（局部性原理）、概念、特征，实现虚拟存储器的关键技术（请求调页/段技术、页/段的置换技术）、硬件/软件支持



一、分区存储管理

- 单一连续分配
- 分区分配
  - 固定分区；
  - 可变分区；
  - 可重定位分区

一、分区存储管理

固定分区：

- 分区容量和数目固定不变，大小可不等，每个分区容纳一道作业

可变分区：动态分区，作业装入内存时才建立分区（根据作业的大小）

- 空闲分区的分配算法：
  - 首次适应：空闲区按地址递增顺序排列
  - 最佳适应：空闲区按容量递增顺序排列
  - 最坏适应：空闲区按容量递减顺序排列

可重定位分区：重定位寄存器、紧凑技术

一、分区存储管理

- 分区的保护：
- 界地址寄存器

  一对界地址寄存器存放分区的上、下界地址，物理地址需满足如下关系：

  - 上界地址寄存器内容<= 物理地址 <=下界地址寄存器内容
- 基址—限长寄存器

  基址寄存器：分区的首址

  限长寄存器：作业地址空间的长度

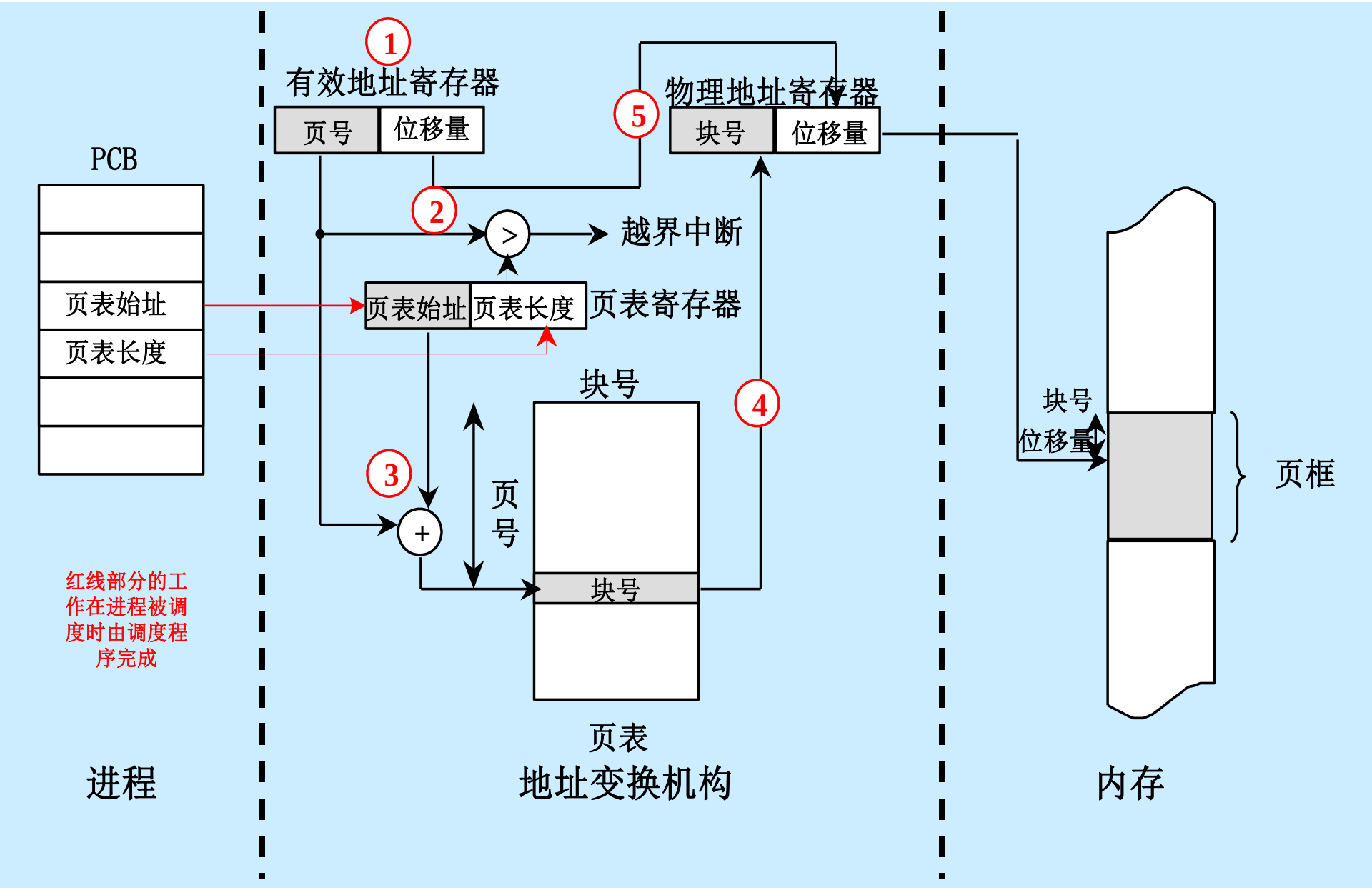
  - 物理地址 – 基址寄存器内容 <= 限长寄存器内容

swapping对换技术

- 把主存中暂时不能运行的进程调出到外存,以便腾出足够的内存空间，再将具备运行条件的进程调入内存；
- 从逻辑上扩充内存空间,从而使整个系统资源利用率提高；
- 整体对换:以进程为单位，进程对换；
- 部分对换:页面对换，分段对换；

二、paging基本分页存储管理

- 引入原因：动态分区方式产生“外碎片”，采用紧凑技术开销大。
- 原理：
  - 进程的逻辑地址空间分成若干个大小相等的片，称为页面或页；
  - 将内存空间分成若干个与页面大小相同的块，称为物理块或页框；
  - 内存的分配以块或页框为单位；
  - 逻辑地址空间和物理地址空间的对应关系由页表实现；
  - 地址转换是在进程执行过程中进行的。


1
有效地址寄存器
页号
位移量
PCB
页表始址
页表长度
2
>
越界中断
页表始址
页表长度
页表寄存器
3
+
页号
块号
块号
页表
4
5
物理地址寄存器
块号
位移量
块号
位移量
页框
红线部分的工作在进程被调度时由调度程序完成
进程
地址变换机构
内存

二、paging基本分页存储管理

- 地址变换机构：完成地址转换
- 将作业的页表放在内存中，而在系统中只设置一个页表寄存器。
- 当一个进程转入执行状态时，其页表的始址和长度从其**PCB**中装入页表寄存器。
- 当进程要访问某个逻辑地址中的指令或数据时，地址变换机构自动地将逻辑地址分为页号和页内地址两部分，并将页号与页表寄存器中的页表长度比较，若页号不小于页表长度，便产生越界中断，否则以页号为索引，去检索页表，从中得到该页的物理块号，送入物理地址寄存器与页内地址拼接，形成物理地址。

二、paging基本分页存储管理

- 快表(TLB)：存放被频繁访问的页面的页表项，设置快表是为了提高内存访问速度，内存访问时间的计算(与快表命中率相关)；
- 多级页表：将页表分页，离散地将各个页表存放到内存块中，从而不需再为页表分配大段连续的内存。
- 哈希页表；
- 倒置页表；

三、Virtual memeory

- 理论基础----程序局部性原理
- 概念：具有请求调入功能和置换功能、能从逻辑上对内存容量加以扩充的存储器系统称为虚拟存储器

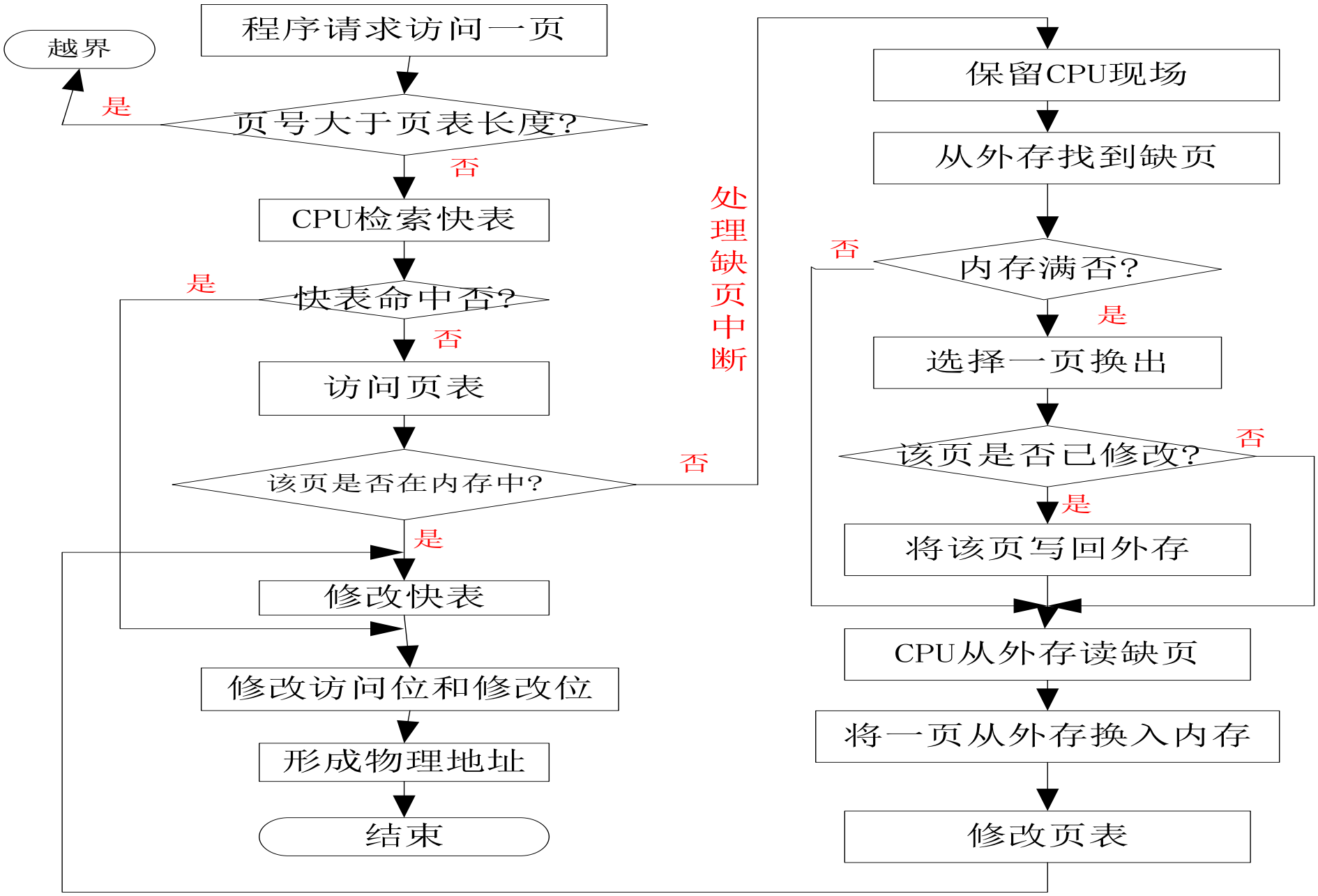
- **Virtual memory is a technique that allows the execution of processes that may not be completely in memory.**
- **Virtual memory is the separation of user logical memory from physical memory.**

三、Virtual memeory

- 虚拟存储器必须建立在离散分配的基础上
  - 请求分页；
  - 请求分段；
- 硬件支持
  - 一定容量的内存；
  - 较大容量的外存；
  - 缺页（段）中断机构；
  - 地址变换机构；

四、demand paging

- 原理：在基本分页的基础上增加请求调页功能和页面置换功能。进程被调度投入运行前，只装入部分页面到内存，其他页在请求时被装入。
  - 页表：增加以下内容：存在位**P**、访问字段**A**、修改位**M**、外存地址。
  - 地址变换过程：
  - 缺页中断处理程序：完成页面的调入。
  - **thrashing**：页面频繁地调入或换出，**CPU**利用率低下



请求分页系统中地址变换的过程

四、demand paging

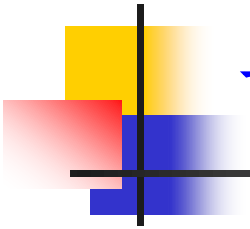
- 页面置换算法：
  - 最佳置换算法（**OPT**）；
  - 先进先出置换算法（**FIFO**）；
  - 最近最久未使用算法（**LRU**）：其近似算法有**CLOCK**算法（一位访问位）、改进型**CLOCK**算法（一位访问位、一位修改位）；

Thrashing Solution

- 局部置换：若一个进程开始颠簸，采用局部置换，不会使其它进程也发生颠簸
- 给进程足够的物理块：根据进程执行的局部模型，来确定进程真正需要多少物理块
- 一种更加直接的防止颠簸的方法是控制缺页频率（**Page-Fault Frequency**）：
  - 颠簸具有较高的缺页率，所以通过控制缺页频率，可以有效地防止颠簸的发生。

存储管理复习要点

- 动态可重定位分区存储管理
- Paging and demand paging
  - 原理、实现、地址变换、页面置换算法、Swapping、thrashing
- Virtual memory
  - 概念、理论基础、实现
- 有关概念
  - 内碎片、外碎片、页表、快表、 Swapping、 thrashing



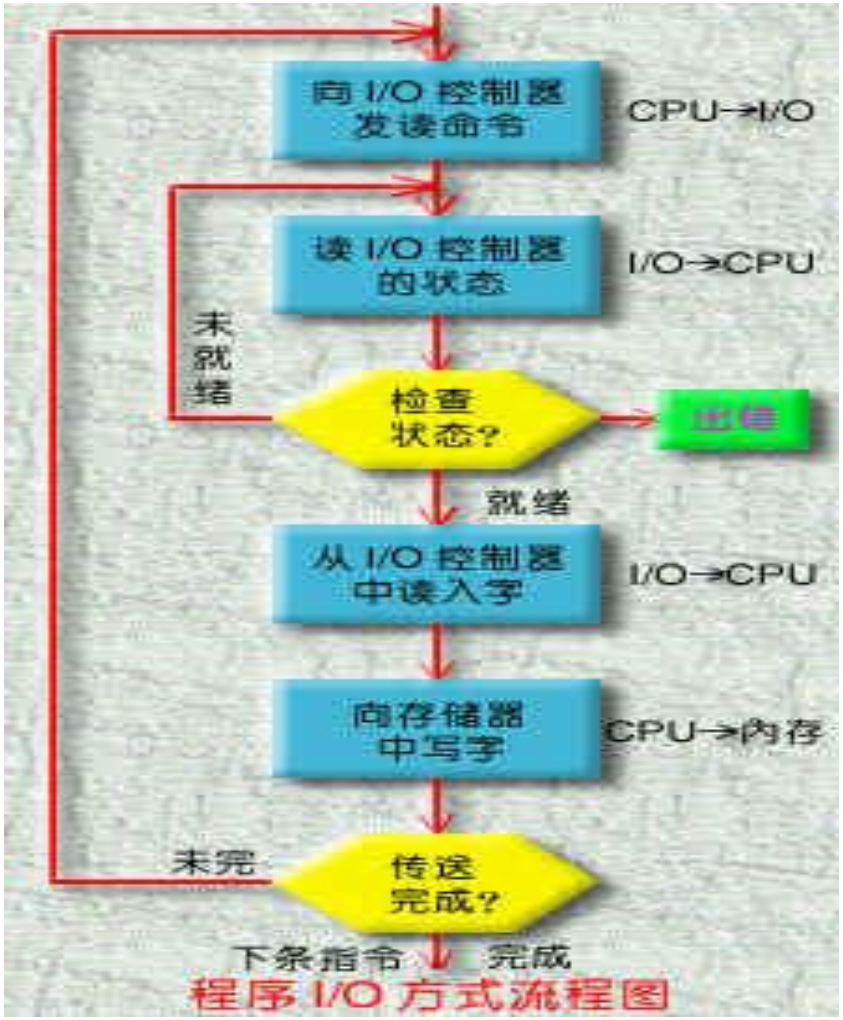
设备管理

- **I/O**控制方式
- 缓冲管理
- 设备分配
- 设备处理
- 磁盘存储器管理

一、I/O控制方式

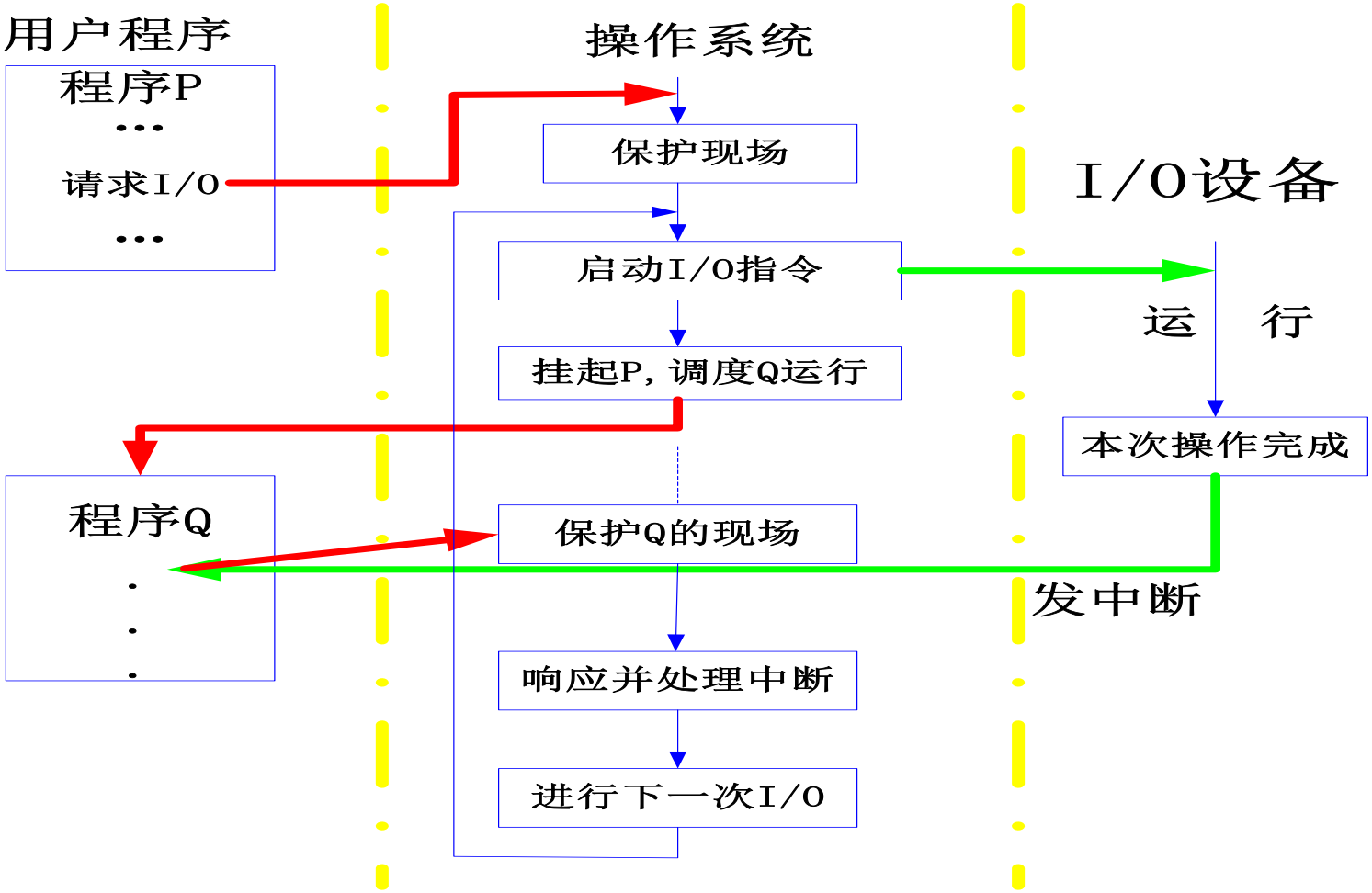
- 宗旨：尽量减少主机对**I/O**控制的干预，把主机从**I/O**控制事务中解脱出来
  - 程序I/O方式（Polling）
  - 中断驱动方式
  - DMA方式
  - 通道方式

程序I/O方式

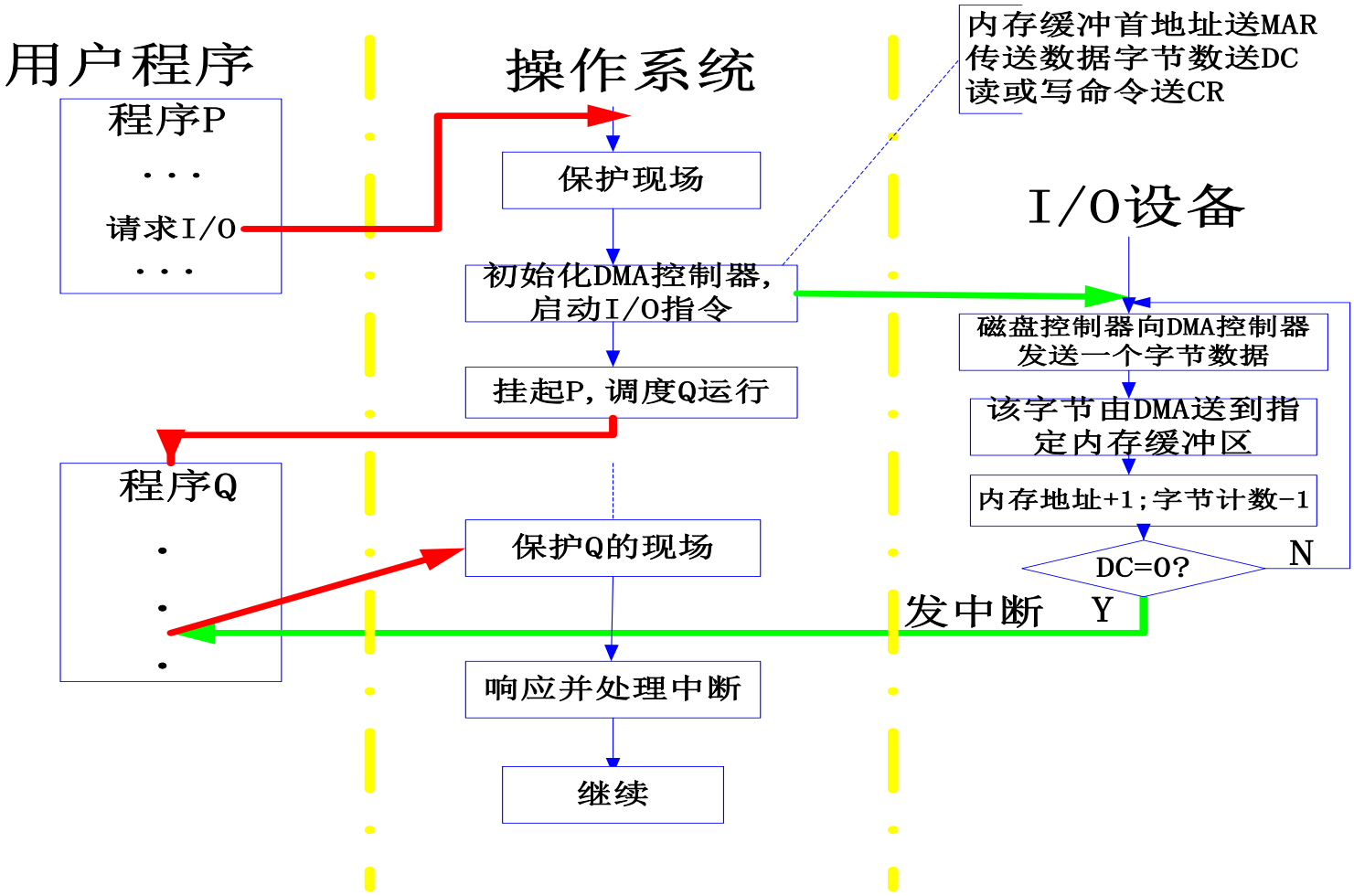


程序 I/O 方式流程图

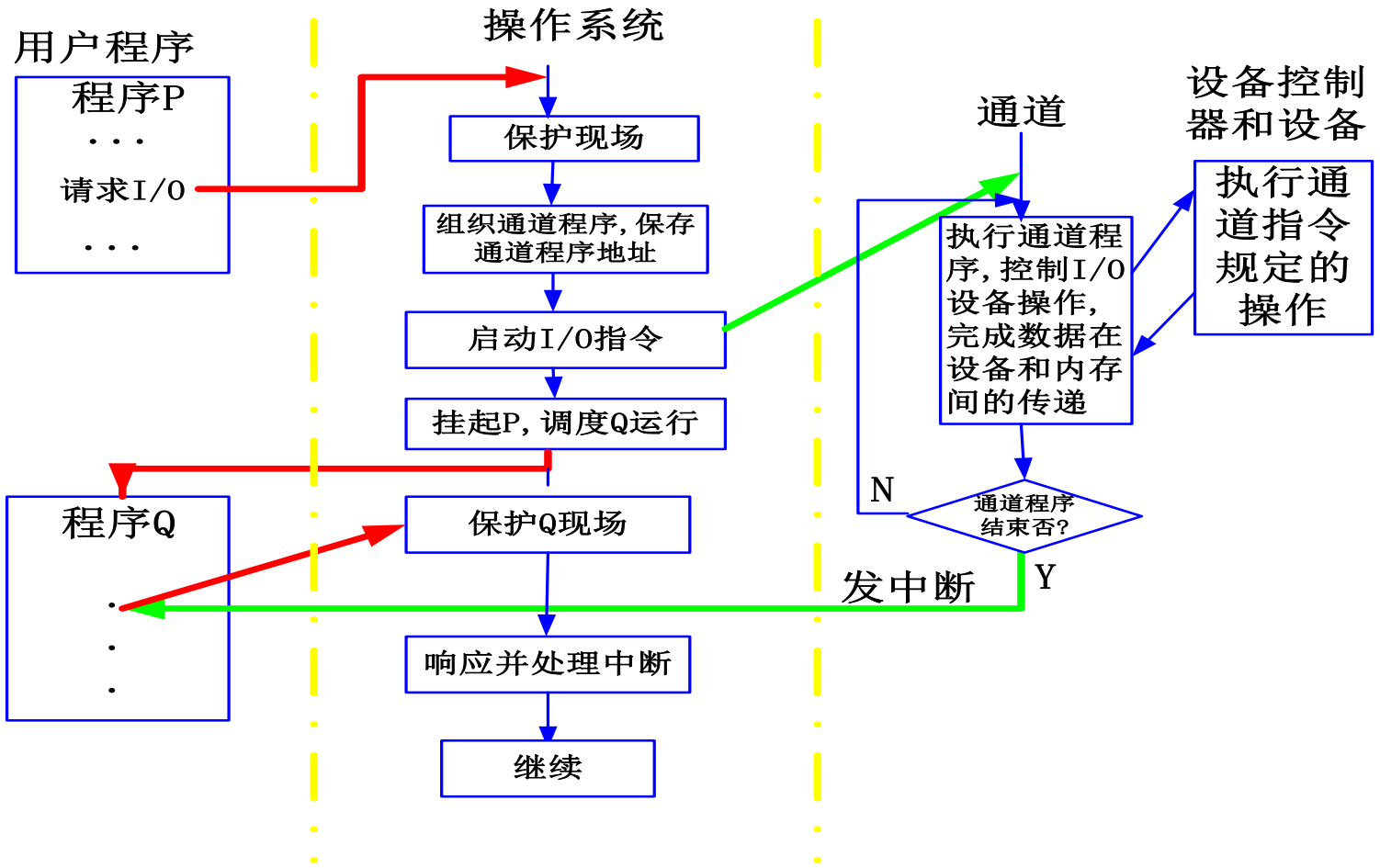
中断驱动方式


用户程序
操作系统
程序P
...
请求I/O
...
保护现场
I/O设备
启动I/O指令
运 行
挂起P，调度Q运行
本次操作完成
程序Q
保护Q的现场
发中断
响应并处理中断
进行下一次I/O

DMA方式


用户程序
操作系统
I/O设备
程序P
· · ·
请求I/O
· · ·
保护现场
初始化DMA控制器，
启动I/O指令
挂起P，调度Q运行
程序Q
·
·
·
保护Q的现场
响应并处理中断
继续
内存缓冲首地址送MAR
传送数据字节数送DC
读或写命令送CR
磁盘控制器向DMA控制器
发送一个字节数据
该字节由DMA送到指
定内存缓冲区
内存地址+1；字节计数-1
DC=0?
N
Y
发中断

I/O通道控制方式


用户程序
操作系统
通道
设备控制器和设备
程序P
· · ·
请求I/O
· · ·
保护现场
组织通道程序，保存通道程序地址
启动I/O指令
挂起P，调度Q运行
执行通道程序，控制I/O设备操作，完成数据在设备和内存间的传递
执行通道指令规定的操作
N
通道程序结束否?
Y
发中断
程序Q
保护Q现场
响应并处理中断
继续

二、缓冲管理

- 为什么要引入缓冲区？
  - 缓和**CPU**和**I/O**设备速度不匹配的矛盾。
  - 降低对**CPU**的中断频率。
  - 提高**CPU**和**I/O**设备之间的并行性，从而提高系统的吞吐量和设备的利用率
- 缓冲池

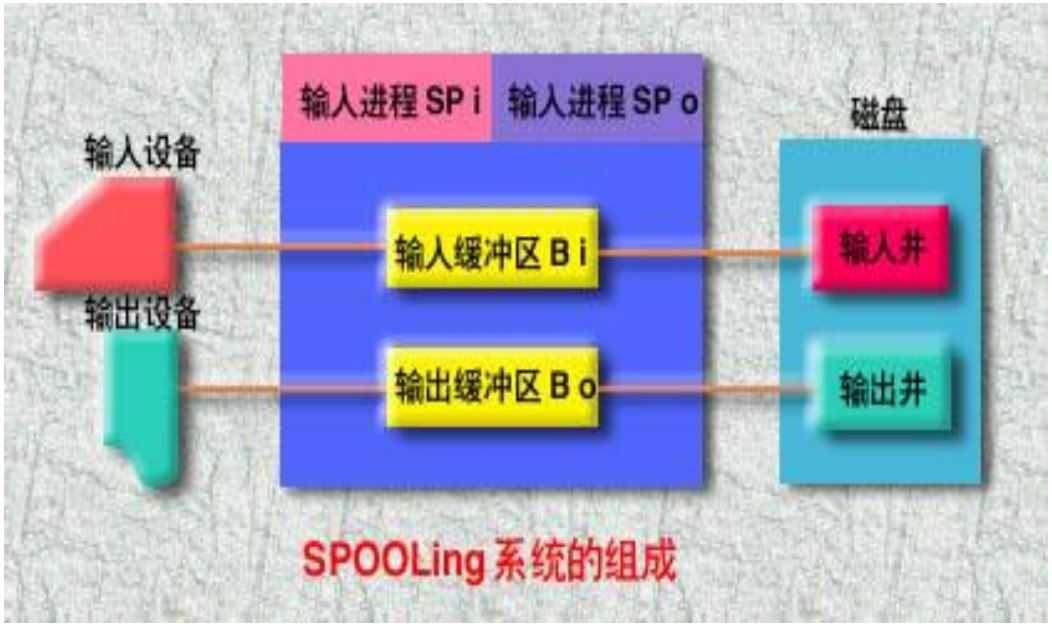
三、设备分配

- 按一定的策略分配设备、控制器和通道
- 数据结构：系统设备表（**SDT**）、设备控制表（**DCT**）、控制器控制表（**COCT**）、通道控制表（**CHCT**）。
- 设备独立性：是指应用程序独立于具体使用的物理设备，它可提高设备分配的灵活性和设备的利用率。
- 为了实现设备独立性，用户程序不直接使用物理设备名（或设备的物理地址），而使用逻辑设备名来请求某类设备；而系统在实际执行时，将逻辑设备名转换为某个具体的物理设备名，实施**I/O**操作。

SPOOLing技术

- **SPOOLing技术**，即同时联机外围操作技术，又称假脱机技术，是指在多道程序环境下，利用多道程序中的一道或两道来模拟脱机**I/O**中的外围控制机的功能，以达到脱机**I/O**的目的，即在联机条件下，将数据从输入设备传送到磁盘，或从磁盘传送到输出设备。通过它可以将一台独占的物理设备虚拟为多台逻辑设备，从而使该物理设备可被多个进程共享。

- **SPOOLing系统组成：**
  - 磁盘上的输入井输出井；
  - 内存的输入缓冲输出缓冲；
  - 输入进程和输出进程。

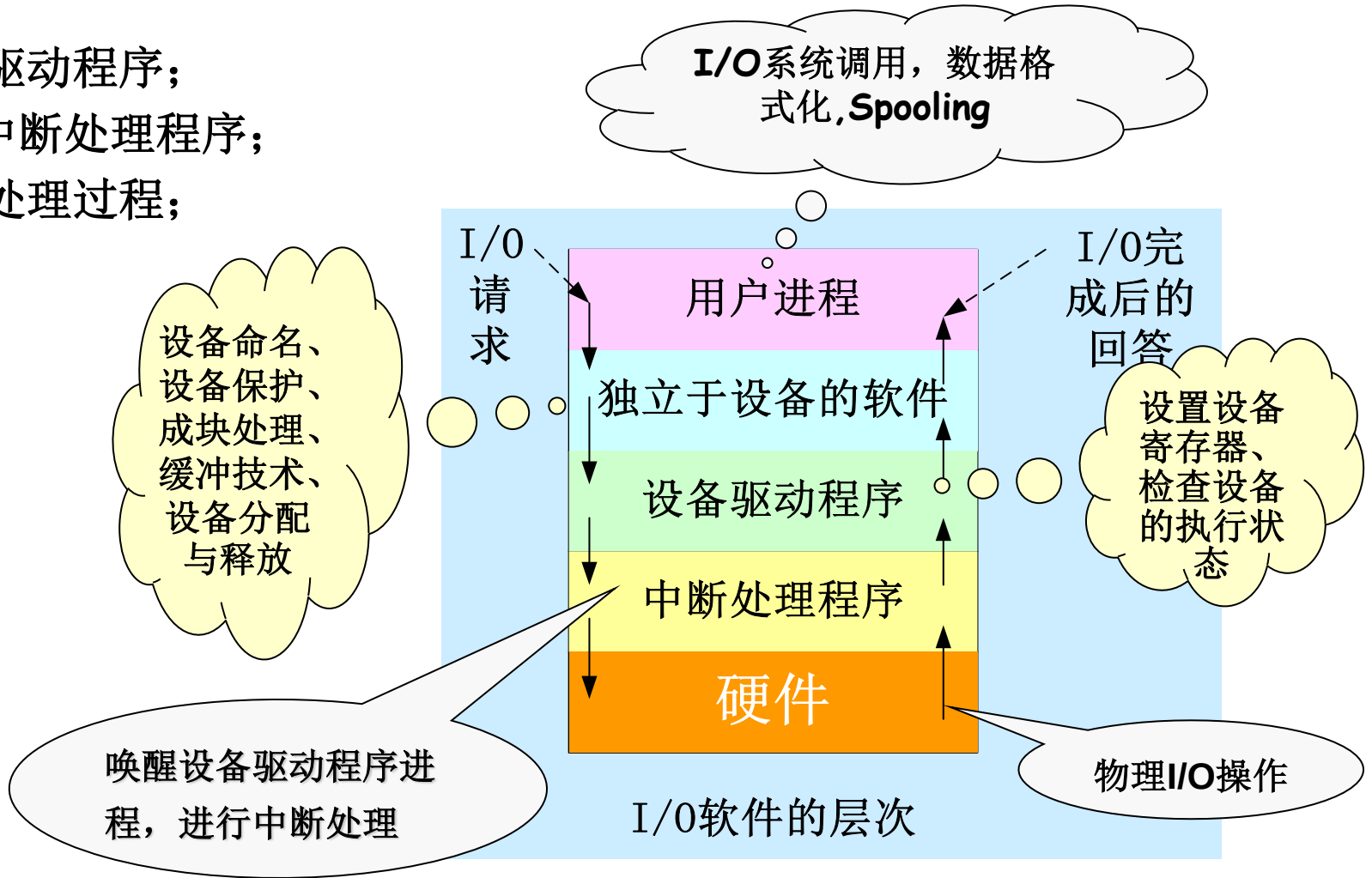


如何利用SPOOLing技术实现多个进程对打印机的共享？

- 答：在利用**SPOOLing**技术共享打印机时，对所有提出输出请求的用户进程，系统接受它们的请求时，并不真正把打印机分配给它们，而是为每个进程做两件事情（**1**）由输出进程在输出井中为它申请一空闲缓冲区，并将要打印的数据送入其中；（**2**）输出进程再为用户进程申请一张空白的用户打印请求表，将用户的打印请求填入表中，再将该表挂到打印队列上。至此，用户进程觉得它的打印过程已经完成，而不必等待真正的慢速的打印过程的完成。当打印机空闲时，输出进程将从请求队列队首取出一张打印请求表，根据表中的要求将要打印的数据从输出井传送到内存输出缓冲区，再由打印机进行打印。打印完成后，再处理打印队列中的下一个请求表，直到打印队列空。

四、设备处理

- 设备驱动程序；
- I/O中断处理程序；
- 设备处理过程；



五、磁盘存储管理

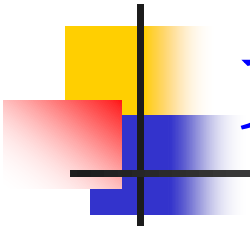
- 磁盘调度——使磁盘的平均寻道时间最短；
- 磁盘访问时间: **seek time, rotational latency, data transmission time**
- **Cylinder, track, sector**
- 磁盘调度调度算法：**FCFS、SSTF、SCAN、CSCAN、LOOK、C-LOOK**
- 提高磁盘**I/O**速度的方法
  - 磁盘高速缓存：在内存中设置缓冲区；
  - 提前读；
  - 延迟写；
  - 优化物理块布局；
  - 虚拟盘（**RAM**）盘

设备管理复习要点

- 设备类型：独占、共享、虚拟；
- **I/O**控制方式；
- 缓冲区：单缓冲、双缓冲、循环缓冲、缓冲池；
- 设备分配数据结构；
- 设备独立性、虚拟设备；
- **SPOOLing**；
- 磁盘调度的优化目标、算法

设备管理

- 名词
  - **Polling**、中断、**DMA**、通道、**SPOOLing**、磁盘调度、缓冲池
- 简答
  - **I/O**控制方式、设备独立性、如何实现虚拟设备



文件管理

- 文件和文件系统
- 文件的逻辑结构
- 文件的物理结构
- 文件存储空间的管理
- 目录管理
- 文件的共享
- 文件系统的可靠性

一、文件和文件系统

- 文件：具有文件名的一组相关信息的集合。
- 文件系统：OS中文件管理有关的软件及被它们管理的文件和文件属性的集合。
- **The file system consists of two distinct parts:**
  - **A collection of files: storing related data**
  - **A directory structure: organizing and providing information about all the files in the sytem**
- **The major task for the OS is to map the logical file concept onto physical devices such as magnetic tape or disk.**
- 文件的使用：用户通过文件系统提供的系统调用来实施对文件的操作。
  - 创建、删除、写、读、打开、关闭等

二、文件的逻辑结构

- 有结构文件（记录式文件）、无结构文件（流式文件）。
- 记录式文件：
  - 顺序文件：记录通常是定长的，顺序存取；
  - 索引文件：记录通常是变长的，方便直接存取；
  - 索引顺序文件：前二者的结合，减少了索引表所占的空间；

三、文件的物理结构

- 文件在外存上的分配通常以块为单位，有三种方法：
- 连续分配
  - 优点：简单、顺序访问速度快、支持随机存取
  - 缺点：外碎片、空间利用率低、不利于文件的动态增长
- 链接分配（离散分配方式）
  - 隐式链接：消除了外碎片、允许作业动态增长；可靠性差、只适于顺序访问；
  - 显式链接（**FAT**）：不支持高效的随机存取，**FAT**表占用空间；
- 索引分配（离散分配方式）
  - 每个文件建立一张索引表，指出分配给该文件的所有物理块号；
  - 支持高效随机存取、消除了外碎片、允许文件动态增长；但索引表占用较多空间；
- 混合索引分配方式（**UNIX**采用）**---i**结点中有**13**个地址项：直接地址（**0-9**）、**1**次间址（**10**）、**2**次间址（**11**）、**3**次间址（**12**）。

四、文件存储空间的管理

- 空闲表法：适用于连续分配。系统建立一张空闲表，每个表项对应一个空闲区，登记的该区的起始块号和块数等。分配算法：首次适应、最佳适应等。
- 空闲链法：把空闲块组织成一个链接文件
- 位示图法：适用于所有分配方式
- 成组链接法：将一个文件卷的所有空闲盘块按固定大小（如每组**100**块）分成若干组，并将每组的盘块数和该组所有盘块号记入前一组的最后一个备用块内，第一组的盘块数（可小于**100**）和该组所有的盘块号记入超级块的空闲盘块号栈中。

四、文件存储空间的管理

- 成组链接法（续）：
- 分配：当系统要为用户分配盘块时，若第一组不只一块，则将超级块中的空闲盘块数减**1**，并将空闲盘块号栈栈顶的盘块分配出去；若第一组只有一块且栈顶的盘块号不是结束标记**0**，则先将该块的内容（记录有下一组的盘数和盘块号）读到超级块中，然后再将该块分配出去；否则若栈顶的盘块号为结束标记**0**，则表示该磁盘已无空闲盘块可供分配。
- 回收：回收时，若第一组不满**100**块，则只需将回收块的块号填入超级块的空闲盘块栈栈顶，并将其中的空闲盘块数加**1** ；若第一组已有**100**块，则必须将超级块中的空闲盘块数和空闲盘块号写入回收块中，然后将盘块数**1**和回收块的块号写入超级块中。
- 空闲盘块号栈是临界资源，必须互斥使用（通过锁来实现）

五、目录管理

- 目录管理的目标：实现按名存取文件，提供快速的目录查询手段以提高对文件的检索速度，并能为文件的共享和重名提供方便。
- 文件控制块（**FCB**）：与文件一一对应。基本内容包括文件名、文件的物理地址、文件的逻辑结构、物理结构、文件的长度、存取权限、文件的建立日期、修改日期及时间、连接计数文件主标识符等。
- 目录：**FCB**的有序集合，其中的每个**FCB**叫作一个目录项。
- 提高目录检索效率
  - 将文件名和描述信息分开（如**UNIX**的**iNode**）；
  - 哈希表；

五、目录管理

- 目录结构：文件目录的组织方式
  - 单级目录结构；
  - 两级目录结构：主文件目录、用户文件目录，不提供用户建立目录的手段；
  - 多级目录结构：当前目录（工作目录）、绝对路径、相对路径；
- 目录查询：系统利用用户给定的路径名对目录进行查询，找到对应的**FCB**或索引结点，然后找到具体的文件；

六、文件共享

- 基于索引结点的方式：文件的索引结点中设置一个链接计数字段**i-nlink**，用来表示链接到本索引结点上的用户目录项的数目。
- 利用符号链实现文件共享：建立一个**LINK**型的新文件。

七、文件保护

- 存取控制矩阵（访问矩阵）
  - 用户分类；
  - 文件访问权限划分
- 存取矩阵的实现
  - 按列：存取控制表
  - 按行：存取权限表

文件系统复习要点

- 文件的访问方法与文件结构
- 文件操作的实现
- 文件存储空间的管理
- 文件目录、目录结构、目录检索
- 文件保护的实现

文件系统复习要点

- 名词：**FCB，File directory，MFD，UFD，FAT；**
- 简答：文件的物理结构，文件操作的系统实现，目录检索，文件的访问方式与文件结构之间的关系。