

| 操作系统-复习

题型	分值
填空	10
选择	20
理解分析	20
算法设计	10
应用	35
讨论	5

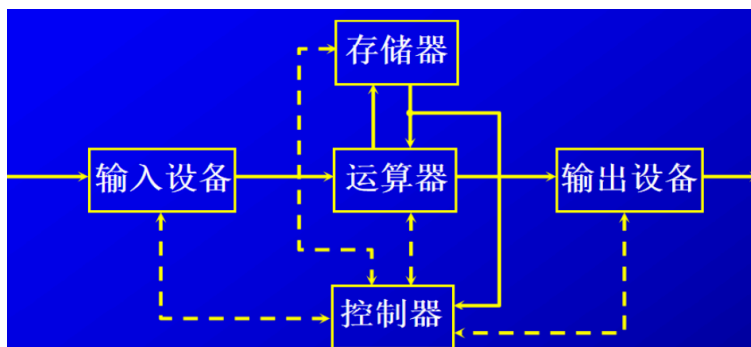
算法设计 —— PV 信号量

应用题 —— 处理机调度、地址变换、存储器扩展

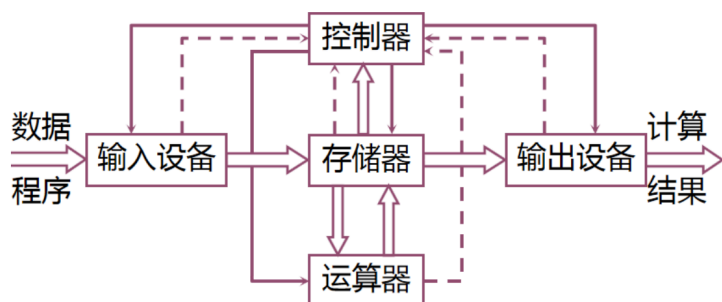
| 第一章 计算机系统概论

| 冯诺依曼体系结构

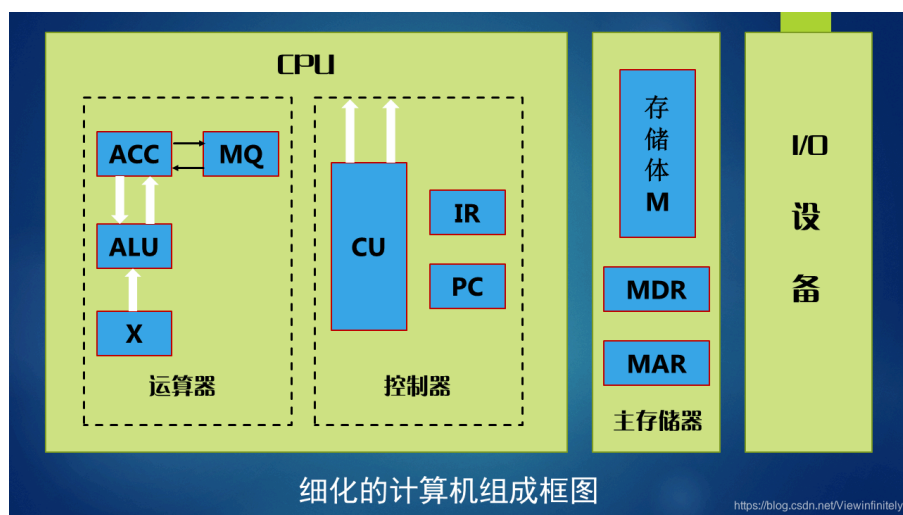
1. 计算机硬件组成包括运算器、控制器、存储器、输入设备、输出设备。
2. 冯诺依曼计算机的核心思想是存储程序，以运算器为中心，输入输出设备与存储器间的数据传送通过运算器完成。
3. 指令和数据以同等地位存于存储器，并可按地址寻访。
4. 指令和数据用二进制表示。
5. 指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置。
6. 指令在存储器内按顺序存放。



7. 现代计算机已转化为以存储器为中心。



各硬件的组成框图和寄存器作用



细化的计算机组成框图

<https://blog.csdn.net/viewinfinitely>

1. **运算器** 细化的组成框图、ACC/MQ/X 寄存器的功能。

1. ACC-累加器
2. ALU-算术逻辑单元
3. X-操作数寄存器
4. MQ-乘商寄存器

	ACC	MQ	X
加法	被加数和		加数
减法	被减数差		减数
乘法	乘积高位	乘数 乘积低位	被乘数
除法	被除数 余数	商	除数

2. **控制器** 细化的组成框图、控制单元CU及PC/IR 寄存器的功能。

1. PC: 存放当前欲执行指令的地址, 具有计数功能 $(PC) + 1$ 。
2. IR: 存放当前欲执行的指令
3. CU: 控制单元

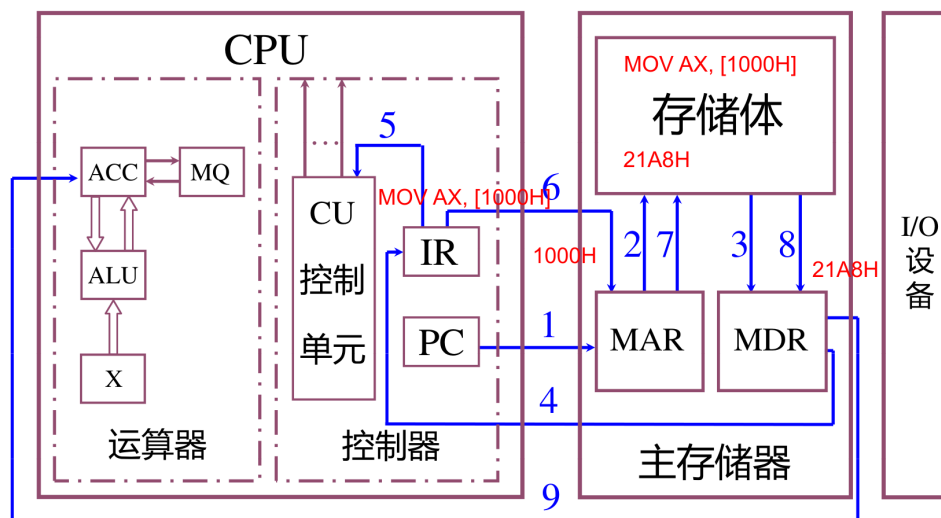
3. **存储器** 细化的组成框图、MAR 寄存器/MDR 寄存器的功能。

1. MAR: 存储器地址寄存器, 反映存储单元的个数。

2. MDR: 存储器数据寄存器, 反映存储单元的存储字长。

设 MAR=4 位, MDR=8 位, 因为 4 位的 MAR 可以存下 2^4 个不同的地址, 则存储单元个数为 16, 存储字长为 8。

结合指令 MOV AX, [4400H], 说明计算机的工作流程。



将内存地址 4400H 中的数据加载到累加寄存器中。

1. 将 PC 寄存器的值送入主存的 MAR 寄存器;
2. 根据 MAR 寄存器, 选通主存的对应单元;
3. 将该单元内容 (即指令 MOV AX, [4400H]) 送入 MDR 寄存器;
4. 将 MDR 寄存器的内容, 送入 IR 寄存器中, 完成取指令操作;
5. 经控制单元 CU 分析, 指令为取数指令;
6. CU 单元将指令的地址码 4400H 送入主存的 MAR 寄存器, 并命令存储器做读操作;
7. 将 4400H 地址单元的操作数送入 MDR 寄存器;
8. 将 MDR 寄存器的内容, 送入运算器的累加器中。

第二章 计算机系统的硬件结构

总线

定义

计算机各部件之间的信息传输线

分类

按传输信息的不同, 系统总线又分为:

1. 数据总线 —— 用来传输各部件之间的数据信息, 双向

2. 地址总线 —— 用来指出操作数在内存的地址或 I/O 设备的地址，**单向**
3. 控制总线 —— 用来发出各种控制信号的传输线，**有出、有入**

判优控制

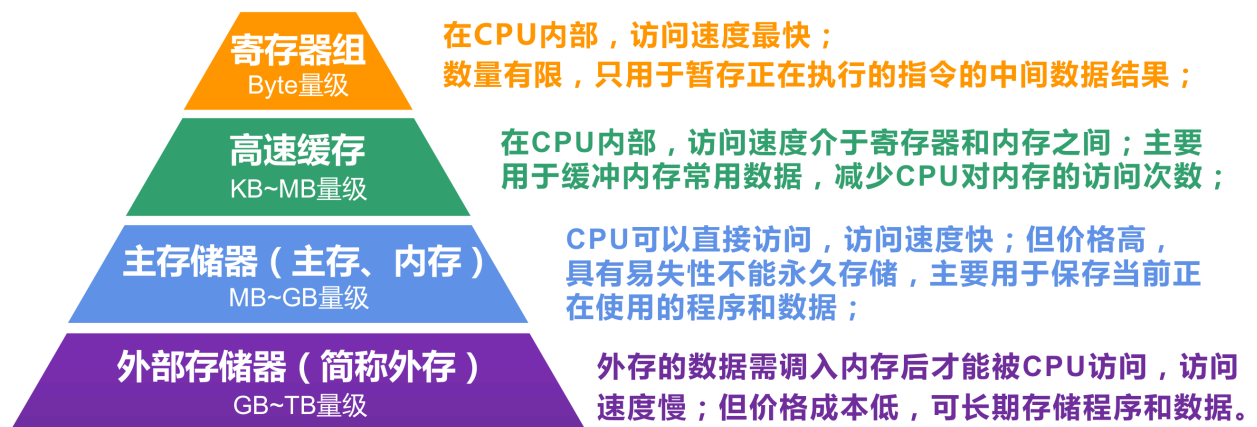
总线判优控制解决多个部件同时申请总线时的使用权分配问题。

分为集中式和分布式

常见的集中式总线控制有3种：链式查询、计数器定时查询、独立请求。

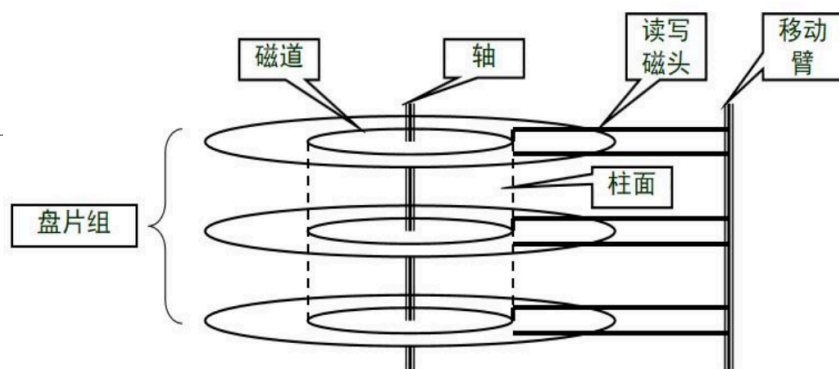
- 链式查询方式连线简单，离控制部件越近的设备优先级越高，易于扩充，**对电路故障最敏感**
- 计数器定时查询方式优先级设置较灵活，对故障不敏感，连线及控制过程较复杂。
- 独立请求方式速度最快，但硬件器件用量大，连线多，成本较高。

存储体系



磁盘

- 磁盘是由表面涂有磁性物质的金属或塑料构成的圆形盘片。
- 数据记录在磁盘中，可以通过磁头来读写。
- 盘面上的数据被组织成一组同心圆，称为磁道。
- 每个磁道有几百个扇区，长度可以固定也可以变化，通常，一个扇区大小固定为 **512** 个字节。



通常，当一个磁道写满后，接着往同一柱面的下一个盘面的磁道上写。

磁盘读写方式：移臂 -> 旋转 -> 读写

寻道时间、旋转延迟时间、传送时间，其中寻道时间、旋转延迟时间称为存取时间，这是达到读或写的位置所需要的时间。

第三章 操作系统引论

概念

1. 定义

操作系统是计算机系统中的一个系统软件，是一些程序模块的集合。

它们能控制和管理计算机系统内各种硬件和软件资源，合理、有效地组织计算机系统的工作，为用户提供一个使用方便、可扩展的工作环境，从而起到连接计算机和用户的接口作用。

2. 五大功能模块：

1. 处理机管理
2. 存储管理
3. 设备管理
4. 文件管理
5. 用户接口

3. 两种用户接口：

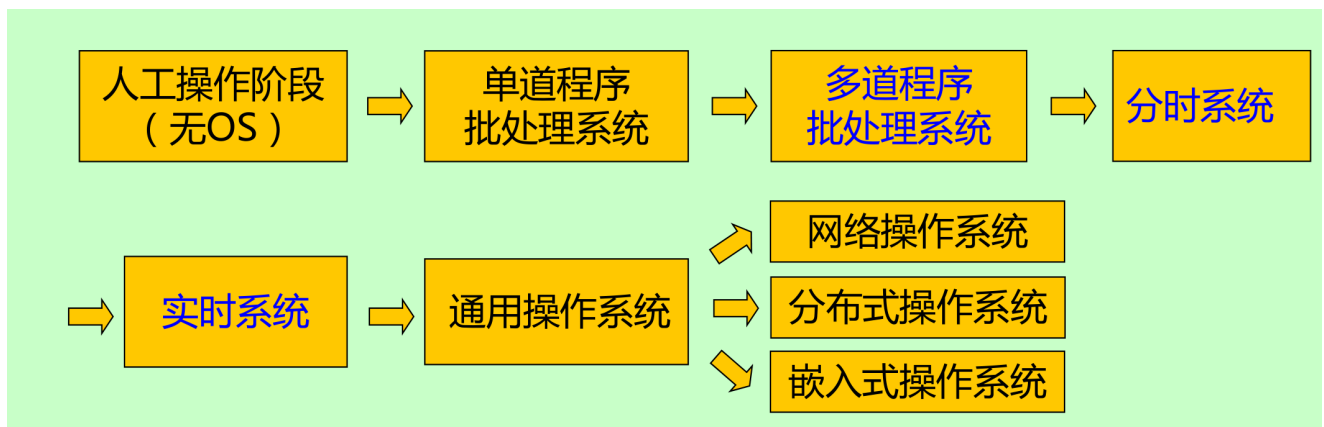
- 命令接口/图形接口 —— 用于计算机操作人员直接操纵和管理计算机
- 程序接口（系统调用） —— 用于在程序中使用操作系统提供的服务

4. 现代操作系统的四大特征：

- 并发性 —— 是指两个或多个事件在**同一时间间隔内**发生，**交替执行**。（单处理机系统）
- 并行性 —— 是指两个或多个事件在**同一时刻**发生，**同时执行**。（多处理机系统）
- 共享性 —— 系统需要对资源共享实施有效管理，避免由于资源竞争不当影响并发执行。
- 虚拟性 —— 通过某种技术将一个物理实体变为若干个逻辑上的对应物的功能。

- 异步性 —— 进程是以人们不可预知的速度向前推进。

发展阶段



1. 单道批处理：实现作业逐个、自动调度运行，内存中仅有一道程序运行，I/O 时也会占用 CPU，导致 CPU 利用率不高。
2. 多道批处理：内存中同时存放若干道程序，轮流使用 CPU，交替执行。宏观上进入内存的几道程序都处于运行状态。
 - 系统资源的利用率提高了
 - 平均作业周转时间长
 - 缺少交互性，用户无法干预调度
 - 需要解决的问题：处理机调度、内存管理、I/O 设备管理、文件管理
3. 分时操作系统
 1. 原理
 - 将处理机的时间分成很短的时间片 (time slice)，以时间片为单位轮流分配给各联机用户（作业）使用。
 - 若某个作业在分配给的时间片内没有完成工作，则该作业暂时中断，等待下一轮。
 2. 特点
 - 同时性：若干用户同时使用系统资源。
 - 独立性：系统中各用户彼此独立操作，互不干扰或破坏。
 - 及时性：时间片往往很短，所以用户能在很短时间内得到系统的响应。
 - 交互性：用户可通过终端方便地与系统进行交互。
4. 实时操作系统
 1. 要求
 - 系统能及时地响应外部请求，在规定时间内完成对该事件的处理。
 - 要求计算机对于外来信息能以足够快的速度进行处理，并在被控对象允许时间范围内做出快速响应，其响应时间要求在秒级、毫秒级甚至微秒级或更小。
 2. 分类
 - 实时控制：把计算机用于机器的自动控制中，计算机要对测量系统所测得的数据及时处理并及时输出。

- 实时事务处理：是把计算机用于飞机订票系统、银行管理系统、情报检索系统等，计算机系统能对用户的服务请求及时做出回答，并能及时修改、处理系统中的数据。

5. 通用操作系统

- 同时兼有多道批处理、分时处理、实时处理的功能，或其中两种以上的功能。
- UNIX —— 通用的、多用户、分时交互型的操作系统
- 单用户多任务操作系统 —— Windows 只允许一个用户上机，这个用户允许运行多个程序。
- 多用户多任务操作系统 —— UNIX, Linux

第四章进程管理

进程是资源分配的基本单位，线程是处理机调度和分派的基本单位。

进程是程序在一个数据集合上的运行过程。

线程是进程内的一个相对独立的、可独立调度和指派的执行单元。

每个线程有对应的线程控制块来记录线程的状态、寄存器的值、堆栈指针。

线程的创建、撤消和调度比进程更节省时间，减小了并发执行的时间和空间开销。

原语指由系统态下执行的某些具有特定功能的程序段，执行过程中不允许被中断。进程控制都是用原语来实现。

进程控制块 PCB

进程控制块 (PCB) 是**系统感知进程的唯一标识**。

记录了 OS 所需的、用于描述进程的当前状态以及控制进程的全部信息。

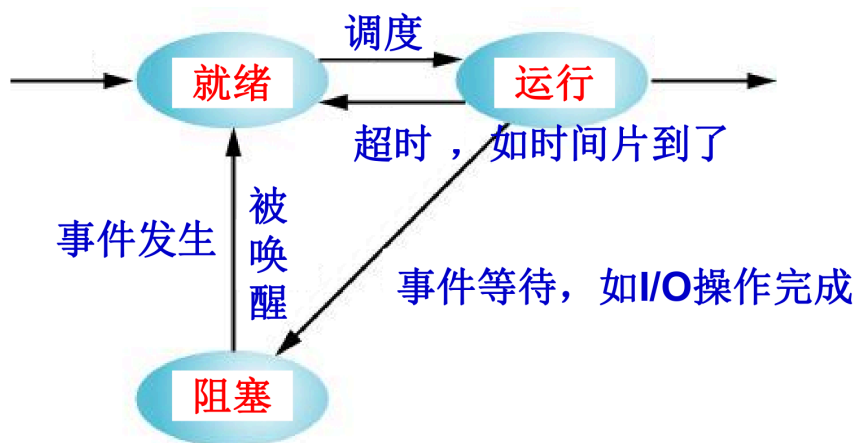
创建进程，首先要创建 PCB，进程与 PCB 是一一对应的。

作用：

PCB 将程序与其运行时的状态信息关联起来，使一个在多道程序环境下不能独立运行的程序（含数据），成为一个能独立运行的基本单位，一个能与其它进程并发执行的进程。或者说，进程控制块是支持多进程并发执行的关键工具。

状态模型

三状态模型



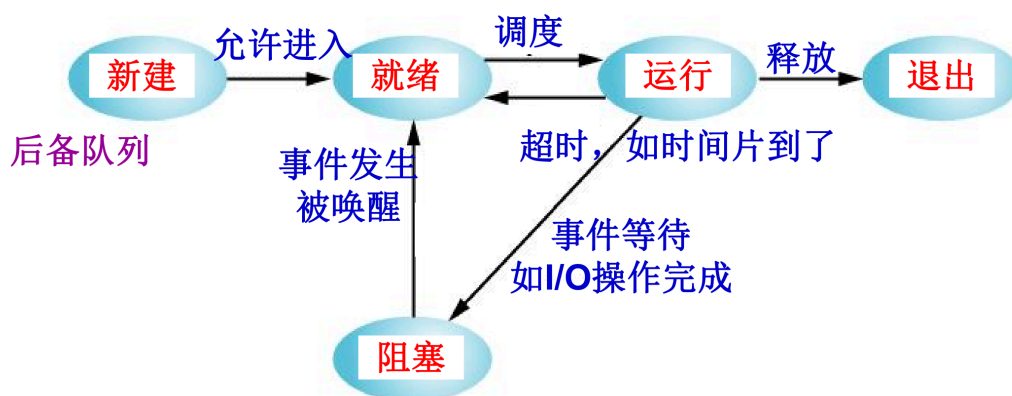
（会用图示加以说明状态的转换原因）

就绪态：该进程做好了准备，只要被调度就可执行。

运行态：进程已经获得必要的资源及 CPU，正在处理机上运行。

阻塞态：进程因某等待某种事件发生而处于暂停执行的状态。

五状态模型



图中未画出的状态转换还有：就绪 -> 退出、阻塞 -> 退出

新建态：刚刚创建的进程。通常 PCB 已经创建，但进程执行的程序代码和相关数据还没有调入内存，即还没有分配地址空间。

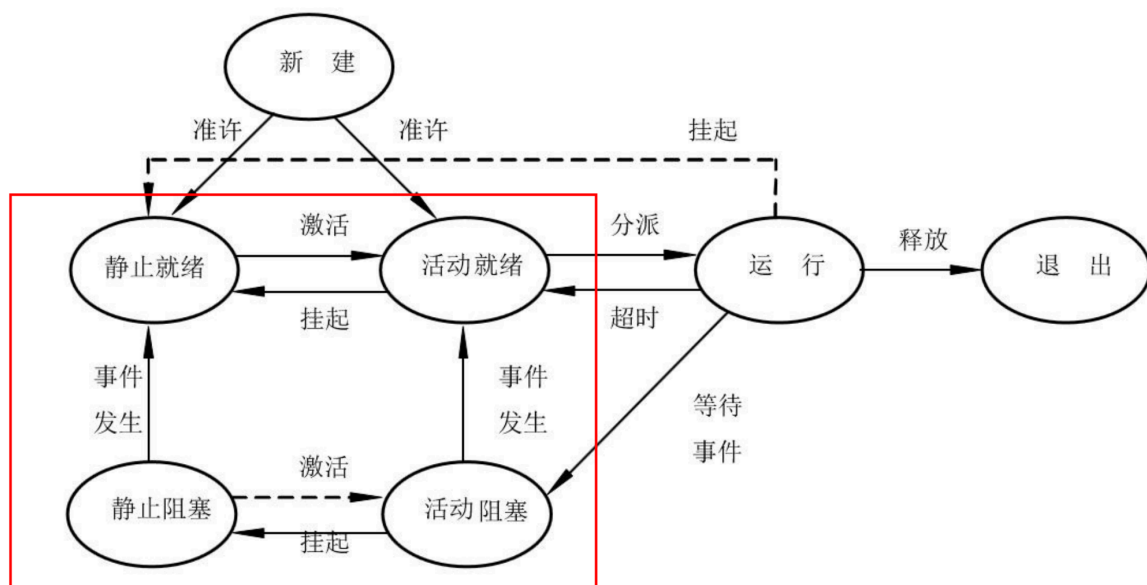
退出态：进程被中止直到释放 PCB 所处的状态。

七状态模型

处理器的速度要比 I/O 的速度快的多。当内存中所有的进程被阻塞时，怎么办？

解决的办法：

主存中某个进程的一部分或者全部内容转移到磁盘（外存）中，这就是挂起。被交换出去的进程称之为挂起状态。



就绪态（活动就绪态）：在主存准备执行。

阻塞态（活动阻塞态）：在主存因等待事件而阻塞。

阻塞挂起态（静止阻塞态）：在外存并等待某事件。

就绪挂起态（静止就绪态）：在外存，但只要被载入内存就准备好执行。

资源的同步与互斥

1. 间接制约关系 —— 资源共享

- 因共享互斥性资源，一个进程在执行期间应具有排斥其他进程的能力。这种行为称为**互斥**。
- 以互斥方式使用的共享资源都称为**临界资源**。
- 每个进程中，访问临界资源的那段代码称为**临界区**。

2. 直接制约关系 —— 进程合作

- 往往是几个进程共同完成一个任务，相互清楚对方的存在及其作用，执行时要有确定的次序。这种因进程合作而产生的制约关系需要进程**同步**。

3. 整型信号量

- 定义：一个整数，代表资源的数目或可同时使用该资源的进程个数。
- 缺点：**忙等**（等待时占用 CPU 资源）

忙等：指的是当资源不足时，进程不断循环检查信号量的值或资源状态，以判断资源是否可用。这种方式会使进程持续占用 CPU 时间。

让权等待：当资源不可用时，进程应该主动释放 CPU 使用权，进入等待队列或挂起状态，避免占用 CPU。

4. 记录型信号量

- 定义：避免忙等的改进信号量，包含两个分量：

◆ 整型变量count

- 代表资源的数目，初始值为非负数。
- 只定义了三个操作：赋初值，Wait、Signal，除赋初值外，其值只能由Wait和Signal修改。

count > 0 : 资源的可用数目
count ≤ 0 : 绝对值表示等待该资源的进程数目。

◆ 进程链表queue：用于链接上述所有的等待进程。

- 特点：
 - 维护等待队列，资源不足时挂起进程。
 - 资源释放时唤醒等待队列中的一个进程。
- 优点：避免忙等，提高效率。

5. 信号量的取值范围

已知信号量初值为 m （反映有 m 个资源可用，或一次可以有 m 个进程共用），如果有 n 个进程共享使用此资源 ($n > m$)，则信号量的取值范围是 $[-(n-m), m]$

6. 生产者-消费者问题

```


const int n = /*buffer size*/
semaphore empty = n;
semaphore full = 0;  semaphore mutex = 1;

void producer()
{ while(true)
  { 生产一个产品
    Wait(empty);
    Wait(mutex);
    将产品放入in指向的缓冲区;
    in = (in + 1) % n;
    Signal(mutex);
    Signal(full);
  }
}

void consumer()
{ while(true)
  { Wait(full);
    Wait(mutex);
    从out指向缓冲区取产品;
    out = (out + 1) % n;
    Signal(mutex);
    Signal(empty);
    消费该产品;
  }
}

void main()
{ parbegin( producer, consumer );
  }
```

能够交换两个P操作？
能够交换两个V操作？为什么？



生产者1 ... 生产者k → [b[1] b[2] b[3] ... b[n]] ← 消费者1 ... 消费者m

out ↑ in ↑

n个缓冲区

沈晓红 · 山东财经大学 | 计算机科学与技术学院

如果交换两个 P 操作，在缓冲区全空，消费者先执行时会产生死锁。

结论：进程中若有多个 Wait 原语，要求同步信号量的 Wait 操作在前，互斥信号量的 Wait 操作在后，以免引起死锁。

第五章处理机调度与死锁

死锁

1. 死锁的定义

- 死锁是指在一个进程集合中的每个进程，都在等待只能由该组进程中的其它进程才能引发的事件，从而无限期的僵持下去的一种局面。如果没有外力作用，它们都将无法

再向前推进。

2. 产生死锁的四个必要条件：

- 互斥条件。
- 请求且保持条件，又称为占有且等待条件。
- 不可抢占条件。
- 循环等待条件。

| 处理机调度算法

1. 处理机调度算法

- FCFS —— 先来先服务调度算法
- RR —— 轮转调度算法
- HRRN —— 最高响应比优先调度算法
- 短进程优先调度算法
- 最高优先级调度算法

2. 指标值

- 周转时间 = 完成时间 - 到达时间
- 响应比 = 周转时间 / 执行时间

| 其他

这一章里面还有死锁的避免，即系统状态的判定——银行家算法。如果处于安全状态，一定不会发生死锁；否则，有可能发生死锁。

| 第六章存储管理

| 物理地址

1. 内存单元的地址称为物理地址，又称为绝对地址。
2. 物理地址是唯一的。物理地址与内存单元具有一一对应的关系。
3. 物理地址的集合称为物理地址空间，又称为绝对地址空间。

| 逻辑地址

1. 与物理地址无关的访问地址称为逻辑地址，或相对地址。
2. 逻辑地址是不唯一的。
3. 逻辑地址的集合称为逻辑地址空间，又称为相对地址空间。

| 地址重定向

根据地址重定位的时机不同，地址重定位又分为：静态地址重定位、动态地址重定位。

1. 静态地址重定向

- 发生在程序装入内存的过程中，在程序运行之前就完成了地址重定位。
- $\text{物理地址} = \text{装入到内存的起始地址} + \text{逻辑地址}$
- 优点：简单，不需要硬件支持
- 缺点：只能装入到内存连续区域；程序不允许在内存中移动

2. 动态地址重定向

- 发生在程序运行过程中，即当执行到某条指令且该指令需要进行内存访问时，再将逻辑地址转换为相应的物理地址。
- $\text{物理地址} = \text{重定位寄存器的值} + \text{逻辑地址}$
- 优点：支持程序在内存的移动；支持程序在内存的离散存储
- 缺点：需要硬件支持

| 分页存储管理

将内存空间划分为大小相等的物理块，将装入模块的逻辑地址空间按物理块的大小划分为大小相等的页。系统为每个进程建立一个页表（数据结构），用于记录页与分配的块的对应关系。

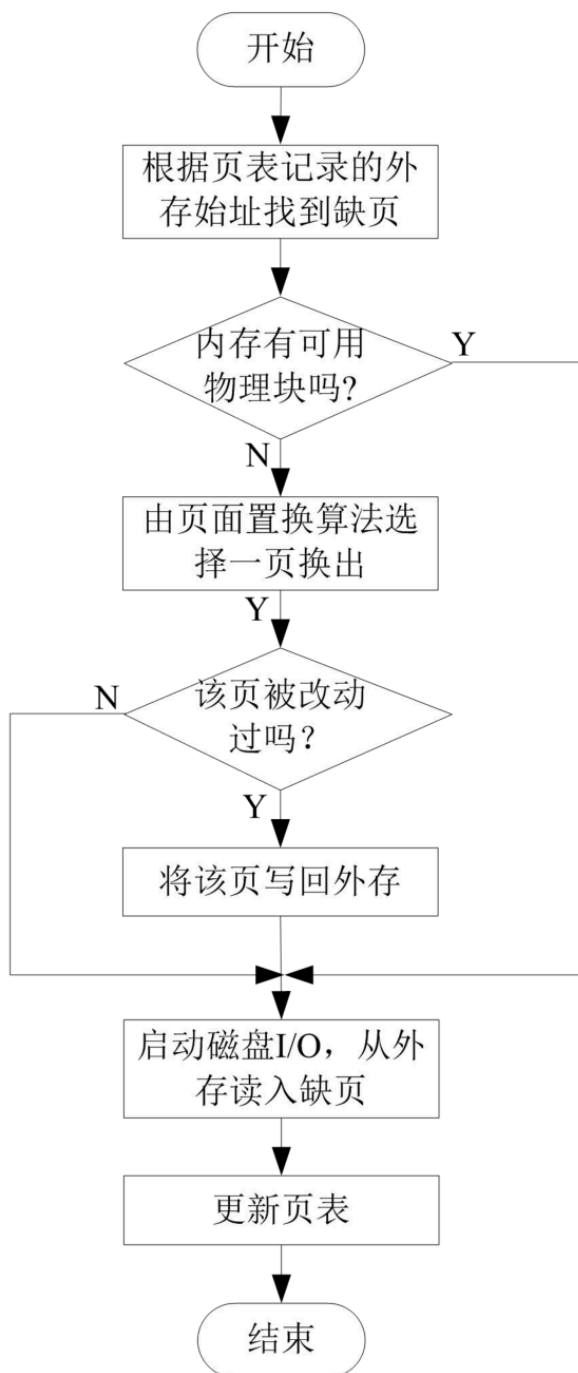
| 内存访问过程

执行读写内存的指令需要访问内存两次：

1. **第一次**：读取页表，进行地址变换，获得物理地址。
2. **第二次**：根据物理地址去内存访问所需要的数据。

为了提高地址变换速度，在地址变换机构中增设一组高速缓冲寄存器，又称为联想存储器，构成一张**快表**，用于存放当前访问最频繁的页表项，可以减少一次对内存的访问。

| 请求分页存储管理



局部性原理

在一段时间内，CPU 总是集中地访问程序中的某一个部分而不是随机地对程序所有部分具有平均访问概率。

抖动现象

刚刚被换出的页，随后又要被访问，造成频繁的磁盘输入输出，降低了系统运行效率，这种现象称为页面抖动。

虚拟内存的定义与特征

定义：

虚拟存储器是指具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储器

系统。

工作原理：

1. **进程创建**： 一个程序要运行时， 仅将那些当前需要执行的部分读入内存， 就为其创建进程并开始执行。
2. **缺页中断**： 进程执行过程中， 如果需要访问的数据或代码不在内存时， 就会产生中断， 操作系统把该进程置于阻塞态， 并取得控制权。
3. **数据调入与置换**： 操作系统负责把该部分内容从外存调入内存。如果此时内存已满， 可以将内存中暂时不用的部分代码或数据置换出去。腾出内存空间后产生磁盘 I/O 读请求， 将缺的代码或数据调入内存。
4. **I/O 请求处理**： 在执行磁盘 I/O 读请求时， 操作系统会分派另一个进程运行。
5. **中断恢复**： 一旦所需的内容读入内存后， 会产生 I/O 中断， 控制权交回操作系统， 从而唤醒原来的阻塞进程转为就绪态， 使其能够继续运行。

| 第七章设备管理

| 外设

外设的概念： 在计算机系统中， 除了 CPU、 内存和系统控制台之外的所有硬件设备。

| 缓冲

1. 作用
 - 缓和 CPU 与 I/O 设备间速度不匹配的矛盾， 提高 CPU 和 I/O 设备之间的并行性。
 - 减少 CPU 的中断频率， 放宽对中断响应时间的限制。
 - 解决数据粒度不匹配的问题。
2. 实现
 - 硬缓冲： 采用专用的硬件缓冲区
 - 软缓冲： 在内存中划出一块专门的空间， 作为缓冲区来存放输入输出的数据。

| SP00Ling

| 概念

虚拟设备技术：

把每次仅允许一个进程使用的物理设备， 改造成能同时供多个进程共享的虚拟设备的技术。

SP00Ling 技术是目前使用最广泛的虚拟设备技术， 组成部分：

- 磁盘上的输入井和输出井
- 内存中的输入缓冲区和输出缓冲区
- 输入进程和输出进程
- 井管理程序

脱机技术：

在外围控制器的控制下，慢速输入设备的数据先被输入到更快速的磁带上，之后主机可以从快速的磁带上读入数据，从而缓解了 CPU 与慢速 I/O 设备的速度矛盾。

假脱机技术：

SPooling 技术借助**多道程序技术**的支持，使用多道程序中的一道程序专门负责输入/输出，将慢速设备的数据写入磁盘，主机随后从磁盘读取数据。

案例：打印机共享

当多个用户进程提出打印请求时，系统会答应他们的请求，但是并不是真正的把打印机分配给他们，而是由假脱机管理进程为每个进程做两件事：

- 在磁盘输出井中，为进程申请一个空闲缓冲区，并将要打印的数据送入其中。
- 为用户进程申请一张空白的打印请求表（包含用户的打印数据存放位置等信息），并将用户的打印请求填入表中，再将该表挂到假脱机文件队列上。

当打印机空闲时，输出进程会从文件队列的队头取出一张打印请求表，并根据表中的要求将要打印的数据从输出井中传送到输出缓冲区，再输出打印机进行打印，这种方式可依次处理完全部的打印任务。

虽然系统中只有一个打印机，但是每个进程提出打印请求时，系统都会同意他的请求，并在输出缓冲为其分配一个存储区，相当于分配了一个逻辑设备，使每个用户进程都能感觉到自己在独占一台打印机，从而实现了打印机的共享。

设备独立性

设备独立性是指应用程序独立于具体使用的物理设备。

为了实现设备独立性，提出逻辑设备和物理设备概念。

应用程序中使用逻辑设备名请求使用某类设备，但执行时，必须使用物理设备。

逻辑设备表

- 实现将逻辑设备名映射为物理设备名。
- 易于实现 I/O 重定向。