操作系统复习笔记

导论

1. 操作系统是管理计算机硬件的程序，它还为应用程序提供基础，并且充当计算机和计算机用户的中介。
2. 计算机系统分为：计算机硬件、操作系统、系统程序与应用程序、用户。
3. 操作系统控制和控制各用户的应用程序对硬件的程序。
4. 事件的发生通常是通过硬件或软件终端来表示
5. 内存是处理器可以直接访问的唯一的大容量存储区域。通常用动态随机访问内存(DRAM)实现
6. IO操作是实际上发生在设备控制器的缓存与设备之间
7. 多处理器系统：增加吞吐量、规模经济、增加可靠性。
8. 操作系统是由中断驱动的
9. 事件总是由中断或陷阱(异常)引起。
10. 作业调度：如果多个作业需要调入内存但没有足够的内存，系统必须在这些作业中做出选择
11. CPU调度：如果有多个任务需要同时执行，那么系统必须做出选择。
12. 用户模式
13. 监督程序模式/管理模式/系统模式/特权模式
14. 程序本身不是进程。进程是一个活动的实体。
15. 单线程进程具有一个程序计数器来明确下一条需要执行的指令。
16. 在任何时候，最多只有一个指令代表进程被执行。
17. 进程是系统工作的单元

概述

1.核心态与用户态（管态与目态）

- 内核态：
  - 处理器、内存、设备等资源管理程序
  - 为应用程序提供良好运行环境的各种原语
  - 文件系统数据及管理
- 用户态
  - 用户数据及管理
- 下列哪几种指令只在核心态下执行（）。

  A 屏蔽所有中断

  B 读时钟日期

  C 设置时钟日期

  D 改变存储映象图

  进程的执行状态分为：核心态和用户态。两者的主要区别就在于进程能否获取计算机的所有资源（核心态可以，用户态则受到限制）。

凡是涉及到计算机根本运行的事情都应该在内核态下执行，而中断、时钟日期、存储映象图都属于系统级（相对应的是用户级）的资源，对这些资源的修改都必须在核心态，但是读取则没有强制要求。

链接：https://www.nowcoder.com/questionTerminal/5ac7387c5dbf4d6fa3f3d62d761058fd?toCommentId=60975来源：牛客网

- 必须在核心态执行的操作
  - 清内存、置时钟、分配系统资源、修改虚存的段表或页表、修改用户的访问权限
  - 输入输出指令（输入输出属于中断操作）
- 从用户态转向内核态
  - 程序请求操作系统服务，执行系统调用
  - 程序运行时产生中断事件（如IO操作完成）
  - 程序运行时产生异常中断（如发生程序性中断，或目态执行特权执行）

    中断和异常是用户态转向内核态的仅有途径
- 从内核态到用户态
  - 计算机通常提供一条称作加载程序状态字的特权指令（Intel x86为 **iret** 指令 ）

2.中断

- 外中断/中断/异步中断： 是指来自处理器之外的中断信号。
  - 时钟中断、键盘中断、它机中断、设备中断
- 内中断/异常/同步中断: 来自处理器内部的中断信号。
  - 访管中断：由执行系统调用而调起
  - 硬件故障中断：电源失效。奇偶校验错误、总线超时
  - 程序性异常：非法操作、页面故障、调试指令、除0、浮点溢出
- 大部分异常发生在用户态，内核态唯一发生的异常是**缺页异常**
- 缺页中断属于I/O中断，是内部中断，发生在内核态

3.三种基础抽象，及引入他们的原因

1. 进程抽象（进程是对进入内存的执行程序在处理器上操作的状态集的抽象）
2. 虚存抽象（虚拟内存的本质是在物理内存的基础上创建一个新的抽象概念，所以虚存是对内存的一种抽象）
3. 文件抽象（文件是设备的一种抽象，通过将文件中的字节映射到存储设备的物理块中来实现文件抽象）
4. 为了方便对物理资源的管理和控制。

4.操作系统分类

1. 按照功能、特点、和使用方式
   - 批处理
   - 分时
   - 实时
2. 操作系统结构分类

1. 单体式
2. 层次式
3. 虚拟机结构
4. 微内核结构

5.系统调用与函数调用的区别

P27

- 调用形式和实现方式不同
- 被调用代码的位置不同
- 提供的方式不同

6. 程序状态字PSW

P58

程序运行时的一组动态信息汇集在一起，称为程序状态字，并存放在处理器的一组特殊寄存器内，以方便系统的控制和管理。

组成：

- 标志寄存器EFLAGS：状态标志、控制标志、系统标志
- 指令寄存器EIP

操作系统结构

1. 进程间通信：共享内存。消息交换技术
2. 向操作系统传递参数的方法：寄存器、内存块(表)传参、堆栈传参
3. 系统调用类型：进程控制、文件管理、设备管理、信息维护、通信

经典同步问题

生产者-消费者问题

```
semaphore mutex = 1;
semaphore empty = n;
semaphore full = 0;
producer(){
    while(1){
        P(empty);
        P(mutex);
        //produce;
        V(mutex);
        V(full);
    }
}
consumer(){
```

```
    while(1){
        P(full);
        P(mutex);
        remove an item from buffer;
        V(mutex);
        V(empty);
        //consume the item;
    }
}
```

爸妈放水果

桌上有一个盘子，每次只能向其中放一个水果；爸爸放苹果，妈妈放橘子，儿子吃橘子，女儿吃苹果。

```
semaphore apple = 0;
semaphore orange = 0;
semaphore plate = 1;
Dad(){
    while(1){
        //prepare an apple;
        P(plate);
        //add an apple;
        V(apple);
    }
}
Mom(){
    while(1){
        //prepare an orange;
        P(plate);
        //add an orange;
        V(orange);
    }
}
Daughter(){
    while(1){
        P(apple);
        //take place am=n apple from the plate;
        V(plate);
        //eat the apple;
    }
}
Son(){
    while(1){
        P(orange);
        //take place an orange from the plate;
        V(plate);
        //eat the orange;
    }
}
```

读者-写者问题

- 读进程优先

```
semaphore rw = 1;
semaphore mutex = 1;
int count = 0;//读者数量
Writer(){
    while(1){
        P(rw);
        // write something;
        V(rw);
    }
}
Reader(){
    while(1){
        P(mutex);
        if(count == 0 ) //从第一个读者进来的时候，锁上共享数据
            P(rw);
        count ++;
        V(mutex);
        //reading;
        P(mutex);
        count--;
        if(count == 0 ) //读者全读结束，释放共享数据
            V(rw);
        V(mutex);
    }
}
```

共享数据只能在没有读者的情况下向写者开放，会导致Writer”饿死“

- 写进程优先

```
int count = 0;
semaphore w = 1;
semaphore rw = 1;
semaphore mutex = 1;
Writer(){
    while(1){
        P(w);
        P(rw);
        //write
        V(rw);
        V(w);
    }
}
Reader(){
    while(1){
```

```
        P(w);//只有当没有写进程的时候，才会开始读
        P(mutex);
        if(count == 0)
            P(rw);
        count++;
        V(mutex);
        V(w);
        //reading
        P(mutex);
        count--;
        if(count == 0)
            V(rw);
        P(mutex);
    }
}
```

当有写进程请求访问时，禁止后续的读进程，等到已在共享文件的读进程执行完毕，立即执行写进程。只有无写进程时，读进程才会再次运行。

哲学家进餐问题

```
semaphore chopstick[5] = {1,1,1,1,1};
Pi(){
        do{
                P(chopstick[i]);
                P(chopstick[(i+1)%5]);
                //eat;
                V(chopstick[i]);
                V(chopstick[(i+1)%5]);
                //think;
        }while(1);
}
```

存在问题：如果所有哲学家同时拿到了左边的筷子，等待右边的筷子，这时候就会出现死锁。

解决方案：

- 至多允许4名哲学家同时进餐。
- 仅左右两边的筷子都可用时，才拿起筷子
- 对哲学家顺序编号，要求奇数好哲学家先拿左边，再拿右边。偶数号相反。

基于第二思路的改进

```
semaphore chopstick[5] = {1,1,1,1,1};
semaphore mutex = 1;
Pi(){
        do{
                P(mutex);
                P(chopstick[i]);
```

```
                P(chopstick[(i+1)%5]);
                V(mutex);
                //eat;
                V(chopstick[i]);
                V(chopstick[(i+1)%5]);
                //think;
        }while(1);
}
```

吸烟者问题

```
semaphore finish = 0;
semaphore offer1 = 0; // 烟草、纸
semaphore offer2 = 0; // 烟草、胶水
semaphore offer3 = 0; // 纸、胶水

producer(){
    while(1){
        random = random();
        if(random%3==0){
            V(offer1);
        }
        else if(random%3 == 1){
            V(offer2);
        }
        else{
            V(offer3);
        }
        P(finish);
    }
}
person1(){
    while(1){
        P(offer1);
        //拿烟草、纸，卷成烟，抽掉
        V(finish);
    }
}
person2(){
    while(1){
        P(offer2);
        V(finish);
    }
}
person3(){
    while(1){
        P(offer3);
        V(finish);
    }
```

```
}
```

交替执行的两个进程

```
P1{
    while(){
            P(自己)
            //
            V(对方)
    }
}
P2{
    while(){
            P(自己)
            //
            V(对方)
    }
}
```

简答题

1. 什么是管程，由哪几部分组成，引入管程的必要性：

一个管程定义了一个数据结构和能为并发进程所执行（在该数据结构上）的一组操作，这组操作能同步进程和初始化并改变管程中的数据。

组成：

- 局部于管程的共享数据结构说明
- 对该数据结构进行操作的一组过程
- 对局部于管程的共享数据设置初始值的语句

管程的引入是为了解决临界区分散带来的管理和控制问题。

2. 进程之间的制约关系

- 同步：是由并发进程之间需要协调完成同一个任务时引起的一种关系。是一个进程等待另一个进程向它发送消息或数据的一种制约关系
- 互斥：并发进程之间竞争系统的临界资源引起的。是一个进程等待另一个进程已经占有的必须互斥使用的资源时的一种制约关系。

3. 三个进程公用一个N个单元的缓冲区，P1每次使用Produce()生成一个正整数并用put()送入缓冲区，P2每次使用getodd() 从中取出一个奇数并用countodd()统计，P3偶数

```
semaphore mutex = 1;
semaphore empty = N;
semaphore odd = 0;
semaphore even = 0;
```

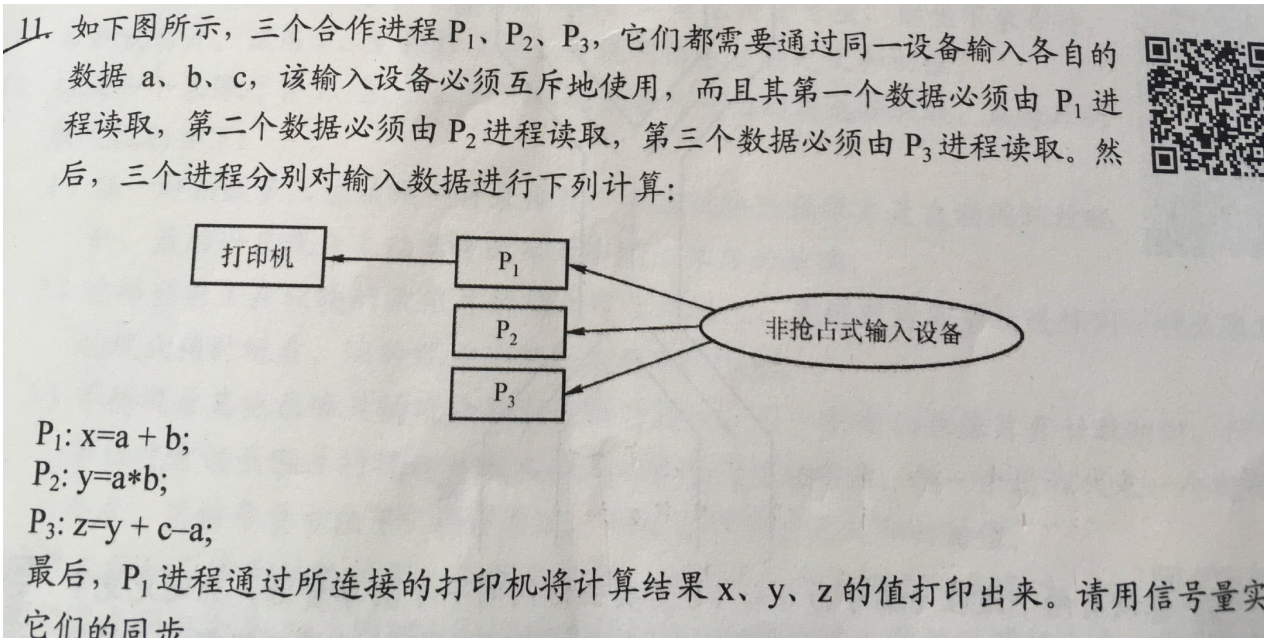
```
Process P1(){
    while(1){
        x = produce();
        P(empty);
        P(mutex);
        Put();
        V(mutex);
        if( x %2 == 0)
                V(even);
        else
            V(odd);
    }
}
Process P2(){
    while(1){
        P(odd);
        P(mutex);
        getodd();
        V(mutex);
        V(empty);
        countodd();
    }
}
Process P3(){
    while(1){
        P(even);
        P(mutex);
        geteven();
        V(mutex);
        V(empty);
        counteven();
    }
}
```

4. 仓库中存放A和B，每次只能存入一种，A产品—B产品 < M ， B 产品 — A 产品 < N

```
//用Sa表示 A与 B还可容纳的数量 的差
semaphore Sa = M-1,Sb = N-1;
semaphore mutex = 1;
process_a(){
    while(1){
            P(Sa);
            P(mutex);
            //add a
            V(mutex);
            V(Sb);
    }
}
process_b(){
```

```
    while(1){
            P(Sb);
            P(mutex);
            //add B
            V(mutex);
            V(Sa);
    }
}
```

5.




```
semaphore S1 = 1;
semaphore S2 = 0;
Semaphore S3 = 0;
Semaphore Sb = 0;
Semaphore Sy = 0;
semaphore Sz = 0;
Process1(){
    P(S1);
    //input a;
    V(S2);
    P(Sb);
    x = a+b;
    P(Sy);
        P(Sz);
    //print x
    //print y
    //print z
}
Process2(){
    P(S2);
    //input b
    V(S3);
```

```
    V(Sb);
    y = a*b;
    V(Sb);
    V(Sy);
    V(Sy)

}
Process3(){
    P(S3);
    //input c
    P(Sy);
    z = y+c-a;
    V(Sz);
}
```

6.某银行提供1个服务窗口和10个供顾客等待的座位。顾客到达银行时，若有空座位，则到取号机上取号，等待叫号。取号机每次只能一个人使用。当营业员空闲时，通过叫号选取一位顾客，并为其服务。

```
semaphore empty = 10;
semaphore full = 0;
semaphore mutex = 1; //互斥使用叫号机
semaphore service = 1;// 互斥进行服务


process customer_i{
    P(empty);
    P(mutex);
    //取号
    V(mutex);
    V(full);
    P(service);//等待叫号
    //被服务
}
process clerk(){
    while(1){
        P(full);
        V(empty);//顾客接受服务时离开座位
        V(service);//叫号
        //服务
    }
}
```

7. 组装自行车

```
semaphore empty_wheel = N-1;
semaphore full_wheel = 0;
semaphore empty_body = N-2;
semaphore full_body = 0;
```

```
semaphore empty = N;
worker1(){
    do{
        //加工一个车架
        P(empty_body);
        P(empty);
            //放入箱子中
        V(full_body)
    }while(1);
}
worker2(){
    do{
        //加工一个车轮
        P(empty_wheel);
        P(empty);
        //放入箱子中
        V(full_wheel);
    }while(1);
}
worker3(){
    do{
        P(full_body);
        //拿一个车架
        V(empty)
        P(full_wheel);
        P(full_wheel);
        //拿俩轮儿
        V(empty);
        V(empty);
        V(empty_wheel);
        V(empty_body);
      //组装成一台车
    }while(1);
}
```

8. 理发问题

- 有一个理发师的椅子，和n个顾客的椅子
- 如果有顾客在椅子上等，那么理发师为他剪发，否则理发师就在自己的椅子上睡觉。
- 如果理发师在熟睡，那么顾客会叫醒理发师，否则顾客会看有没有空椅子，有的话，他坐下等，否则，他将离开理发店。

```
semaphore customer = 0;
semaphore barber = 0;
int chair = n;
int waiting = 0;
semaphore mutex = 1;
customer_i(){
    P(mutex);
```

```
    if(waiting < chair){//等待的人数小于椅子数
        waiting++;//等待的人加一
        V(mutex);//解除锁
        V(customer);//唤醒barber
        P(barber);//等待barber
        get_haircut()
    }else{
        V(mutex);//走人
    }
}
barber(){
    while(1){
        P(customers);//等待customer
        P(mutex);
        waiting--; //等待的人减一
        P(mutex);
        Cut_hair(); //理发
        V(barber); // 唤醒下一个等待的顾客
    }
}
```

- 要通过PV操作来进行进程间信息的传递，A想要唤醒B，则在A中V一个变量，在B中需要唤醒的位置用P等待这个变量

内存管理

内存管理方式	内部碎片	外部碎片	硬件支持	解决碎片方法	解决空间不足	
单道连续分配	√	X	界地址寄存器+越界检查机构		覆盖	
多道固定连续分配	√	X	上下界寄存器+越界检查寄存器 & 基地址寄存器 + 长度寄存器+ 动态地址转换机构			
多道可变连续分配	X	√	上下界寄存器+越界检查寄存器 & 基地址寄存器 + 长度寄存器+ 动态地址转换机构	紧凑		
分页存储管理方式	√	X	页表寄存器+ 基本地址变换机构+ 具有快表的地址转换机构			
分段存储	X	√				
段页式	√	X				

固定的分配产生内部碎片，不固定的产生外部碎片

分段式：每段长度不一样— 不固定

段页式：既有固定，又不不固定，固定为主 —— 内部

名词解释

1. 逻辑地址（相对地址），即用户编程所使用的地址空间，从0开始编号，有两种形式：一维逻辑地址（地址）；二维逻辑地址（段号，段内地址）
2. 物理地址（绝对地址）即程序执行所使用的地址空间，处理器执行指令时按照物理地址进行。

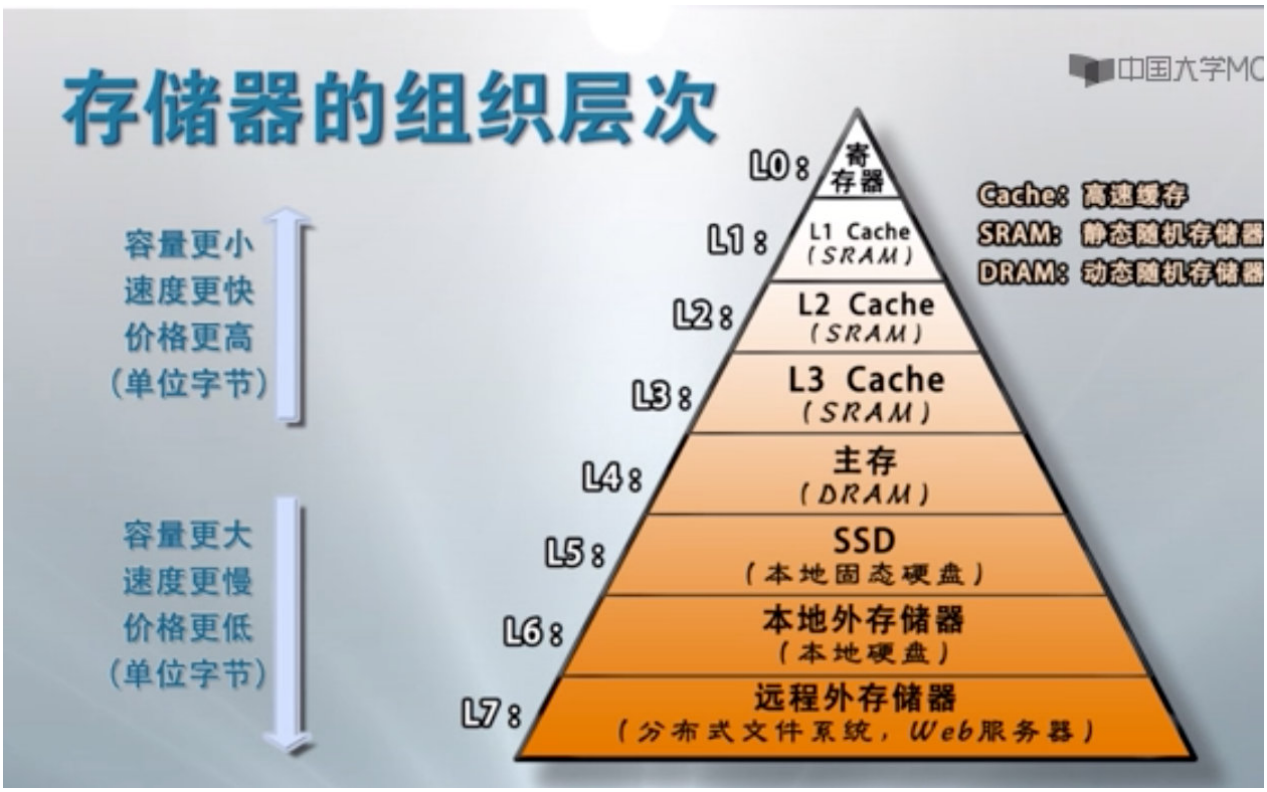
主存储器的复用

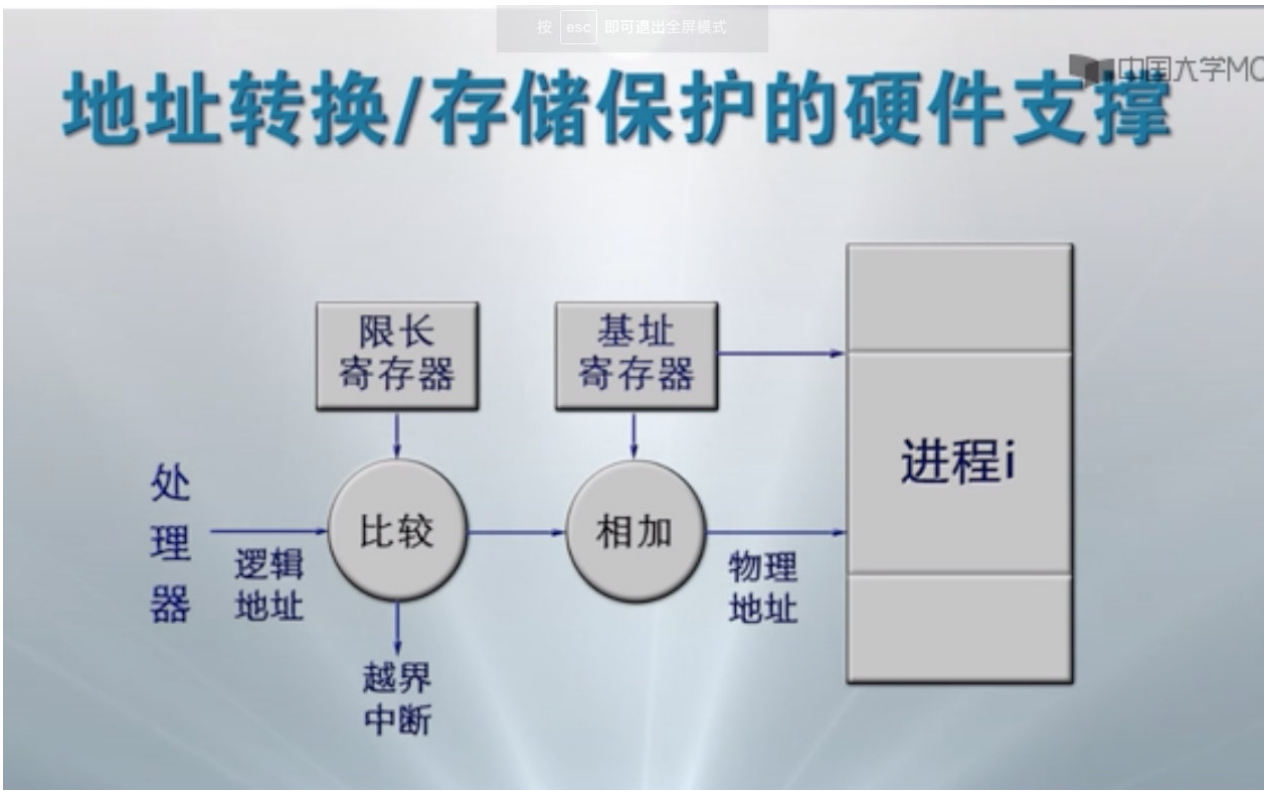
多道程序设计需要复用主存

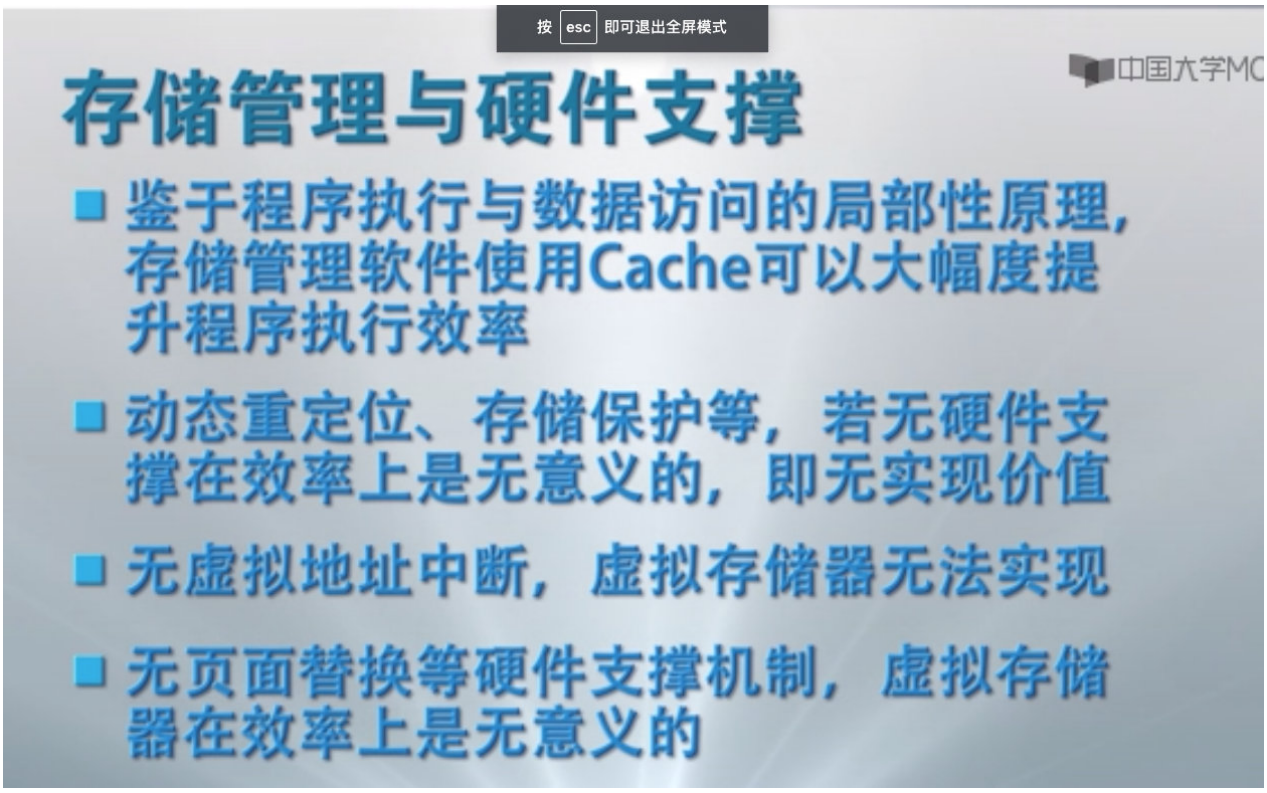
- 按照分区复用：
  - 主存划分为多个固定/可变尺寸的分区
  - 一个程序/程序段占用一个分区
- 按照页架复用：
  - 主存划分为多个固定大小的页架
  - 一个程序/程序段占用多个页架。

硬件支撑：

存储器的组织层次







1. 程序执行过程

地址转换：又称重定位，把逻辑地址转换成绝对地址

静态重定位：在程序装入内存时

动态重定位：在CPU执行层序时，硬件实现

存储保护：为避免主存中多个进程相互干扰，必须对主存中的程序和数据进行保护（软硬件协同完成）

- 私有主存区重点信息：可读可写
- 公共区中的共享信息：根据授权
- 非本仅进程信息：不可读写

2. 扩充内存

1. 对换技术：把部分不运行的进程调出
2. 虚拟技术：只调入进程的部分内容

注意：这一工作需要软硬件协作完成

- 对换进程决定对换，硬件机构调入
- CPU处理到不在主存的地址，发出虚拟地址异常，OS将其调入，重执指令。

3. 连续分配（单连续分区存储管理）

每个进程占用一个物理上完全连续的存储空间（区域）

1. 单用户连续分区存储管理

适用于单用户单任务操作系统

一般采用**静态重定位**。

2. 固定分区存储管理

支持多个分区

分区数量固定

分区大小固定

可用静态/动态重定位

硬件实现代价低

缺点：

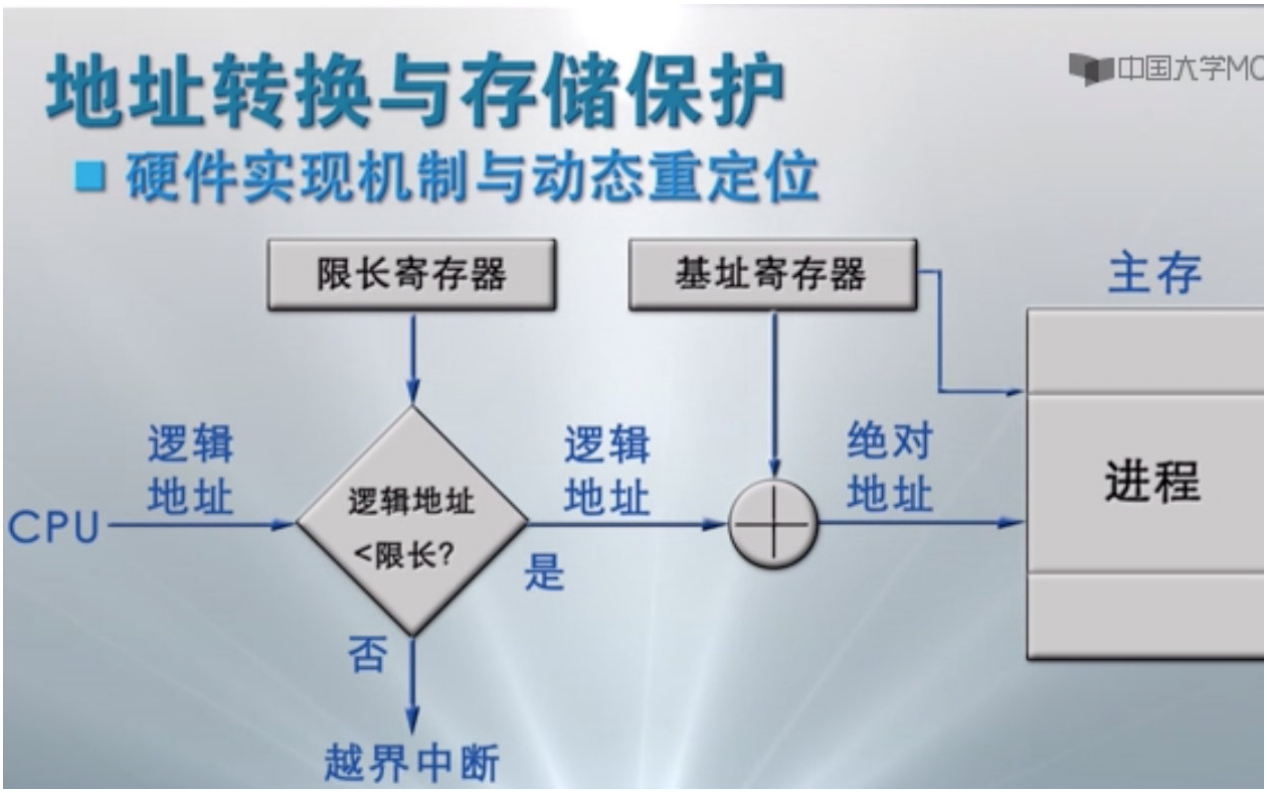
不够灵活，不适用大尺寸程序，又存在内部碎片（内零头）

3. 可变分区存储管理 （动态分区分配/多道可变连续分配）

- 按进程的内存需求来动态划分分区
- 创建一个进程时，根据进程所需的主存量查看主存中是否有足够的连续空闲空间
    - 若有，按照需要量分割一个分区
    - 若无，令该进程等待主存资源
- 由于分区大小按照进程实际需要量来确定，因此分区的个数是随机变化的

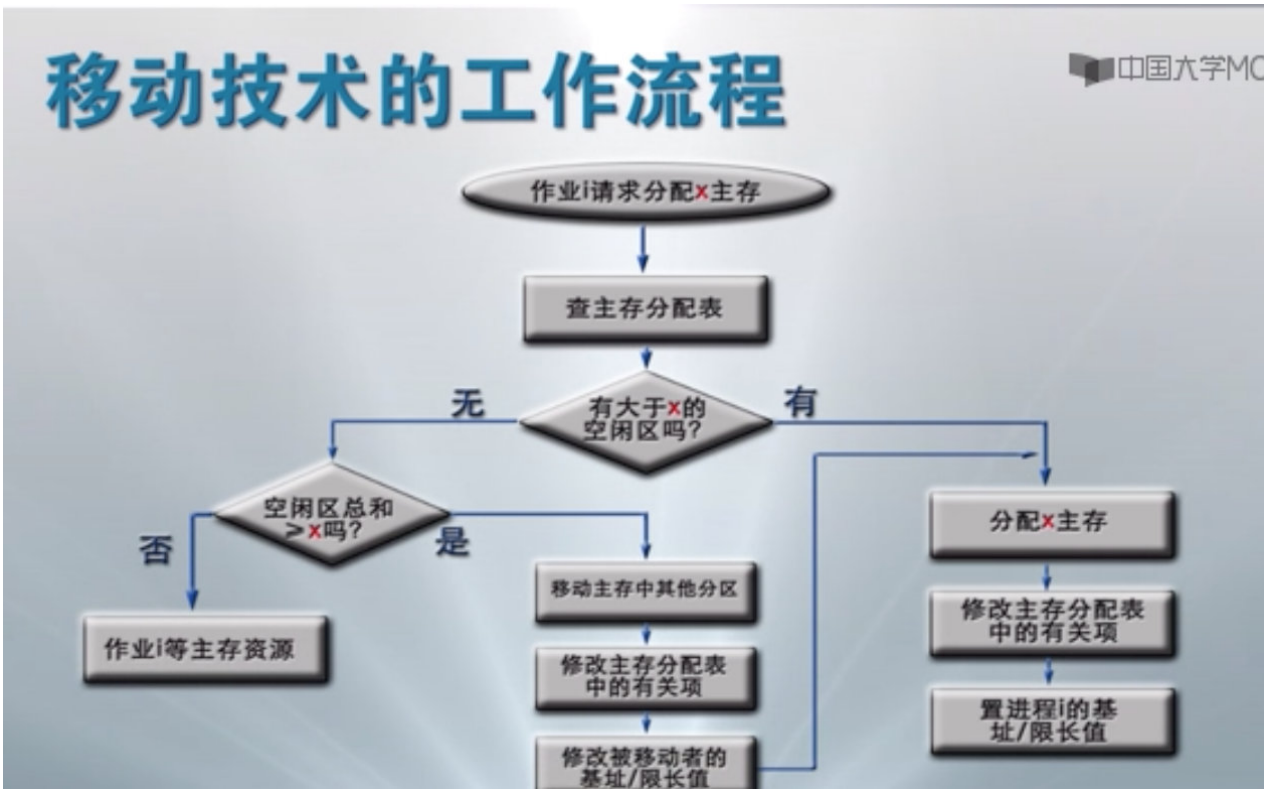
内存分配

- 最先适应
- 邻近适应
- 最优适应
- 最坏适应



移动（紧凑）技术

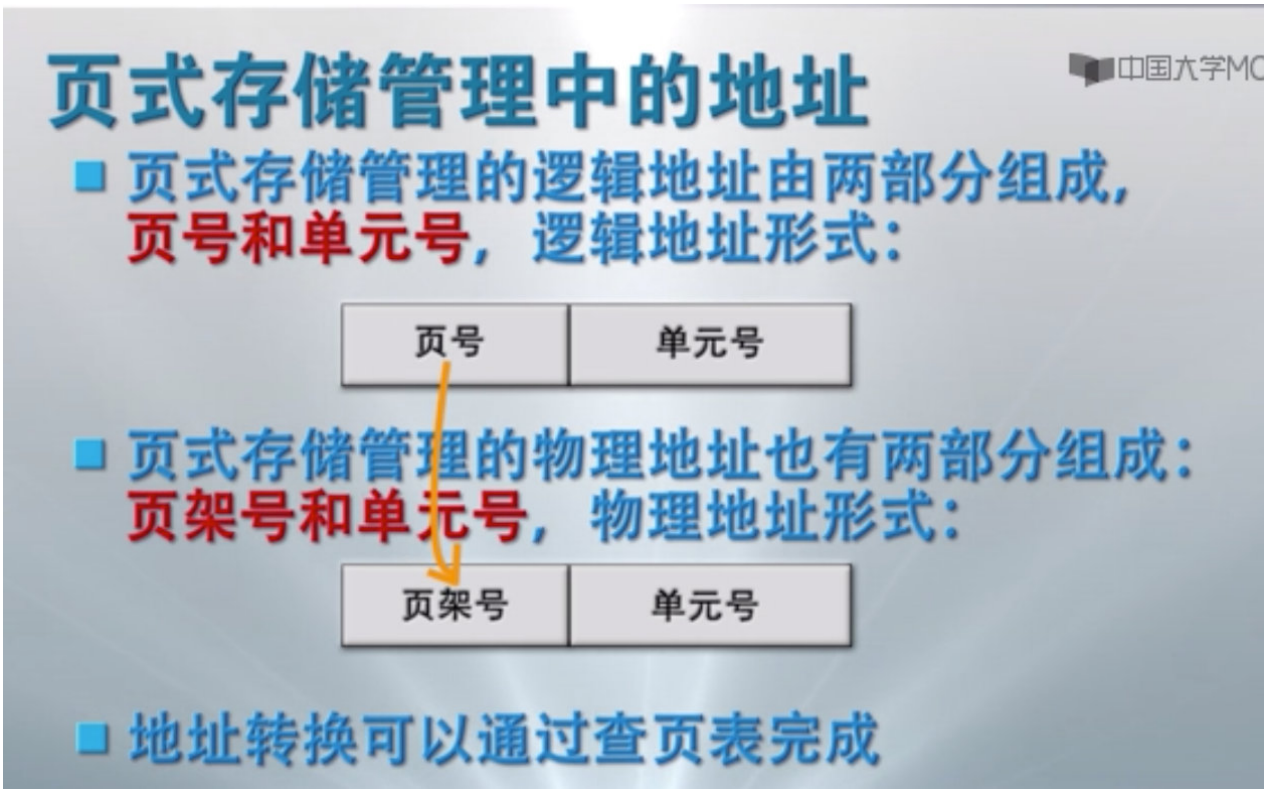
要基于动态重定位

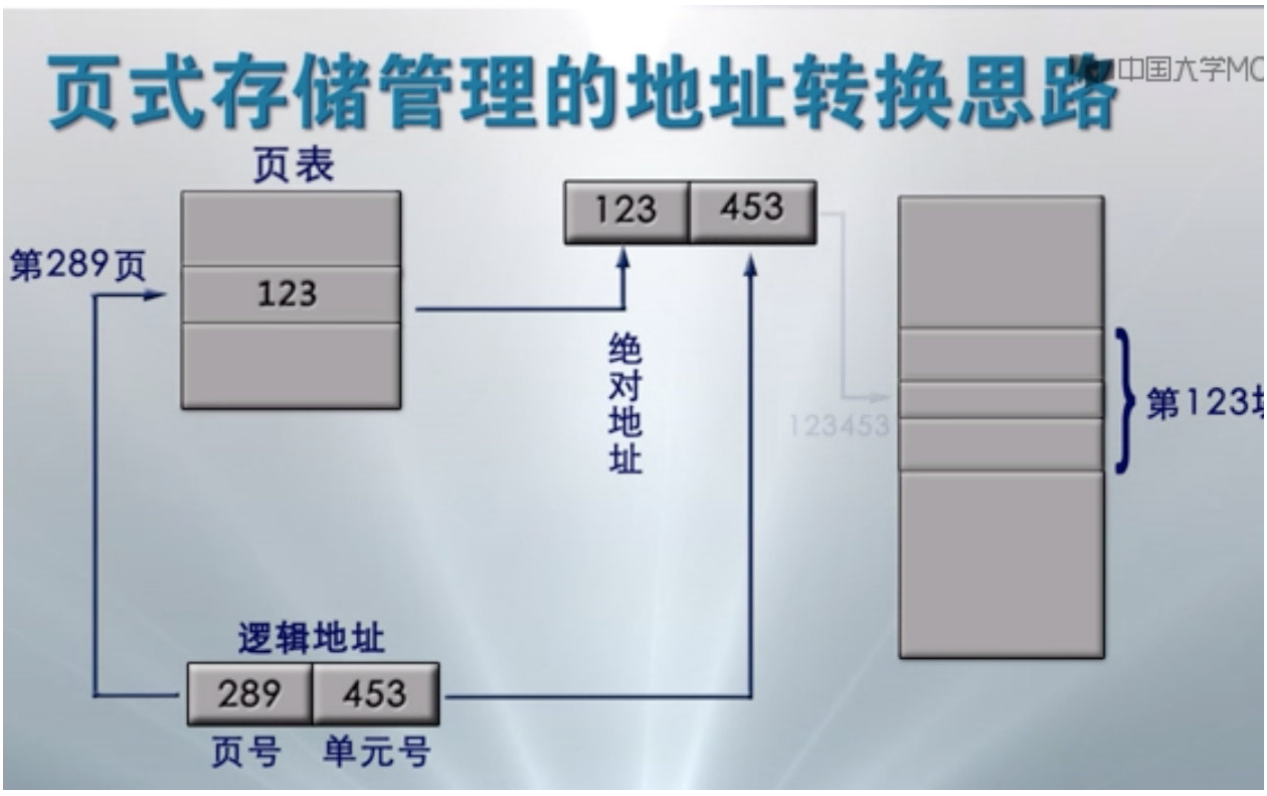


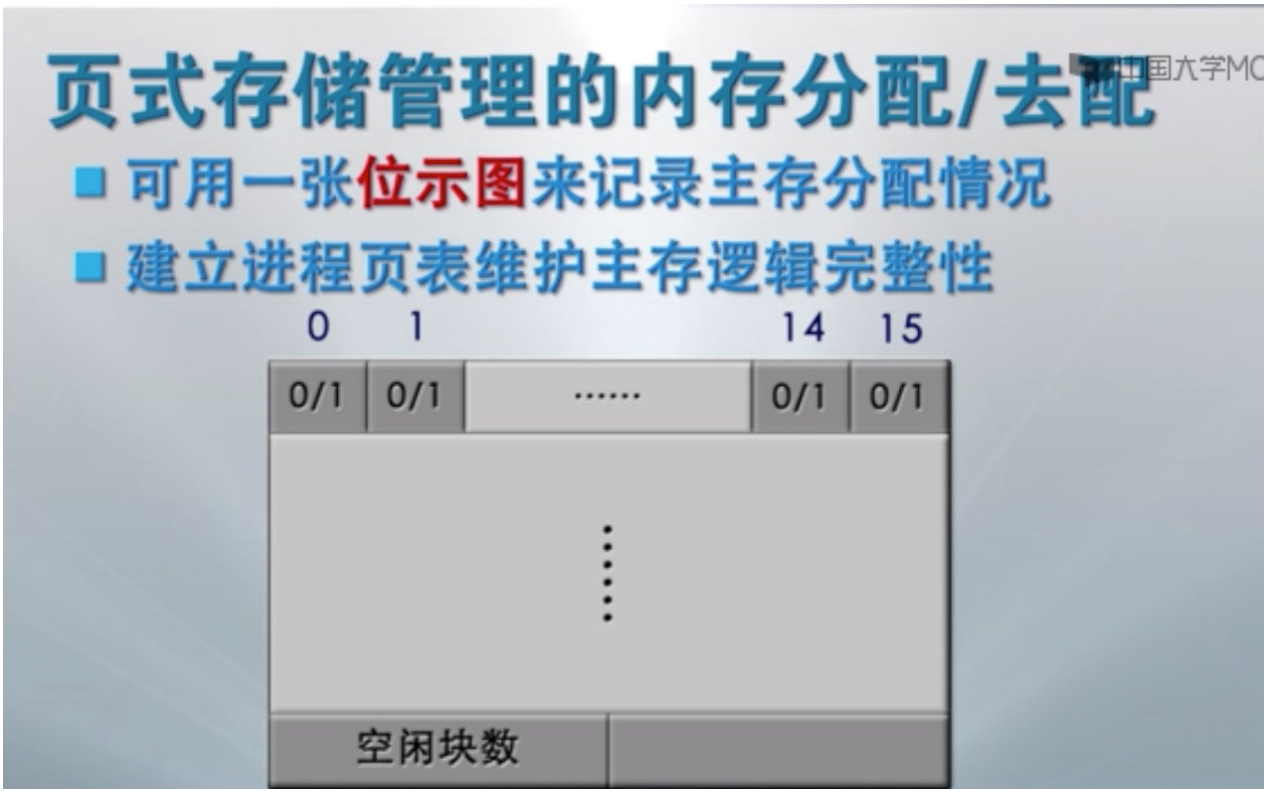
4.非连续分配

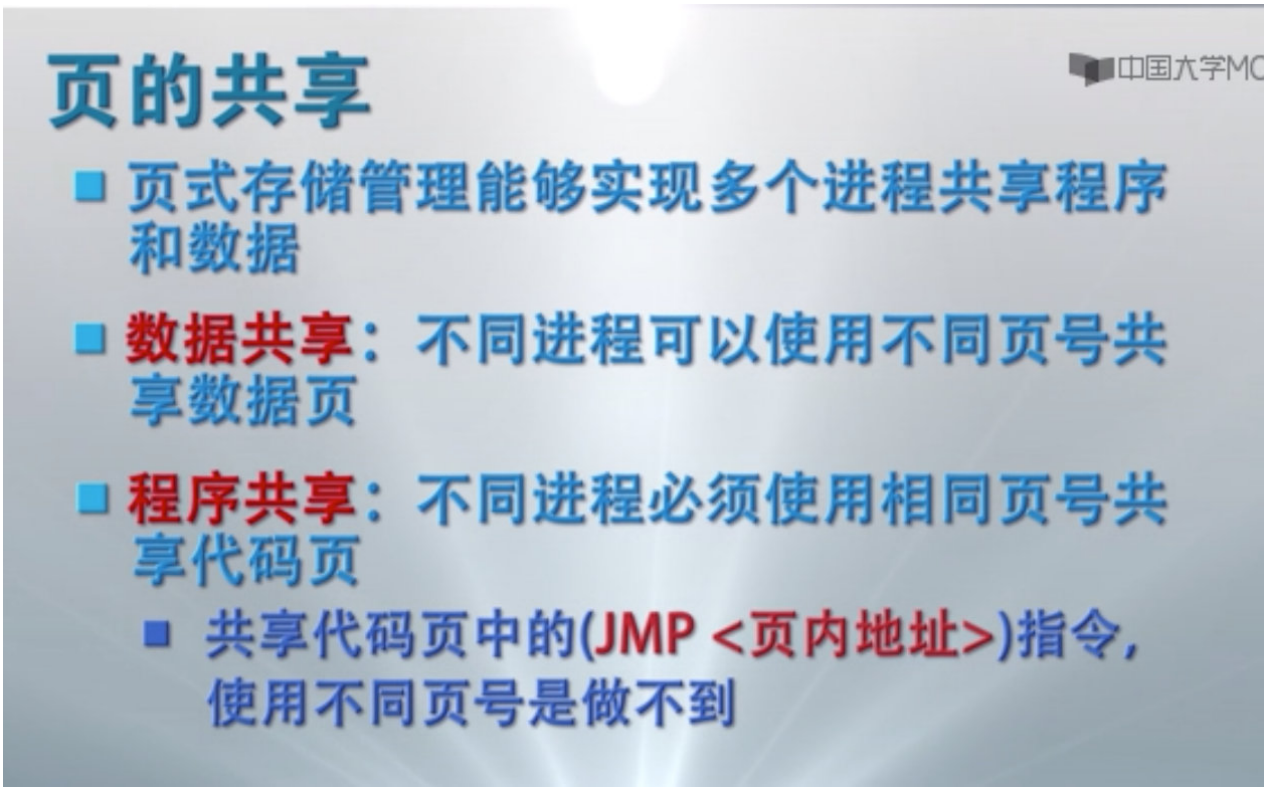
1. 页式存储管理

blog









地址转换

页表放在主存：每次地址转换必须访问两次主存

- 按页号读出页表中的相应页架（框）号
- 按计算出来的绝对地址进行读写

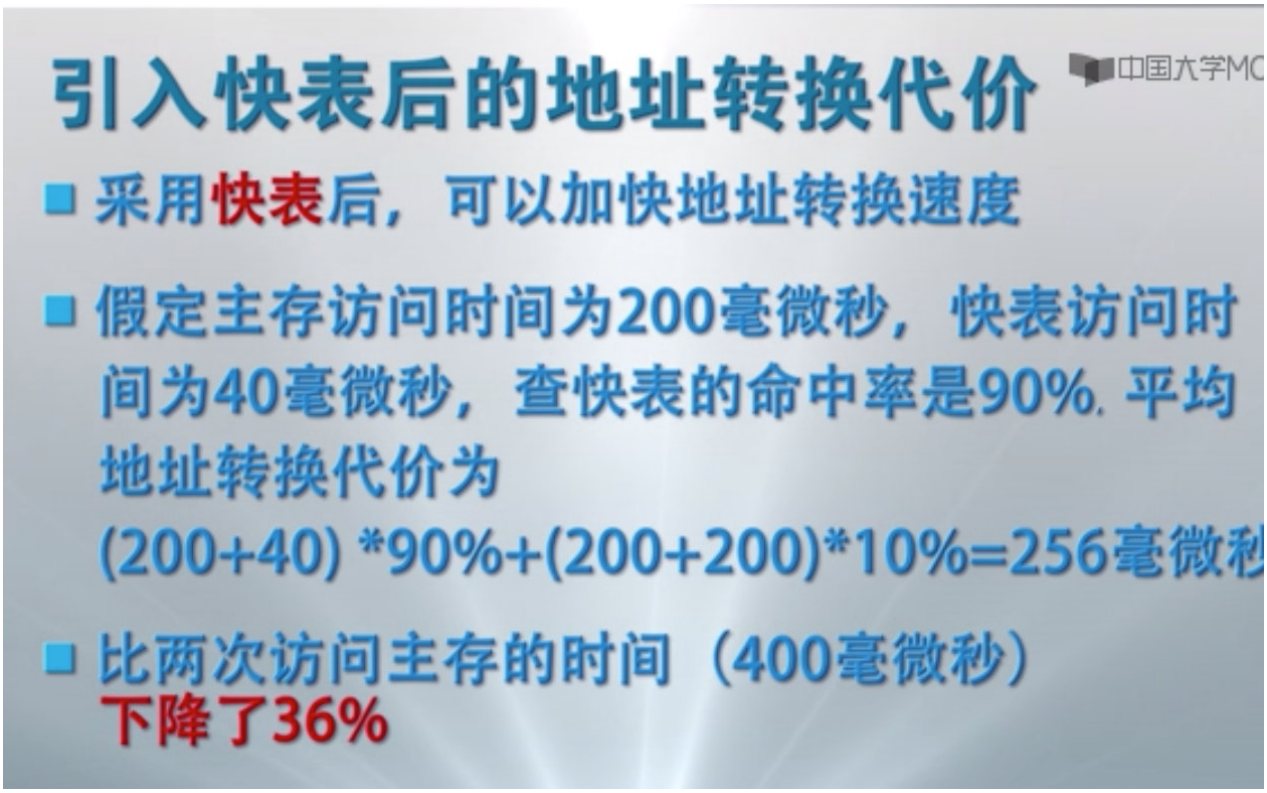
降低了存取速度

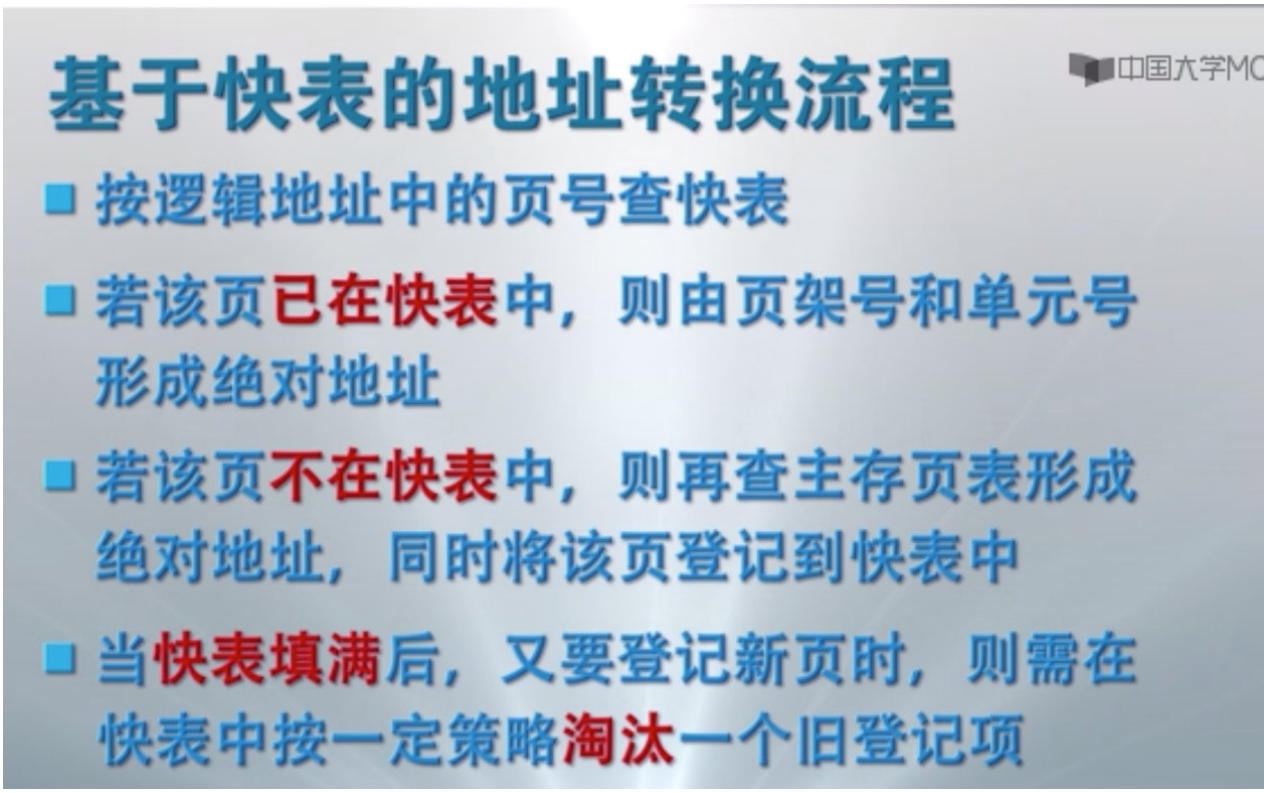
解决方案：利用cache储存部分页表

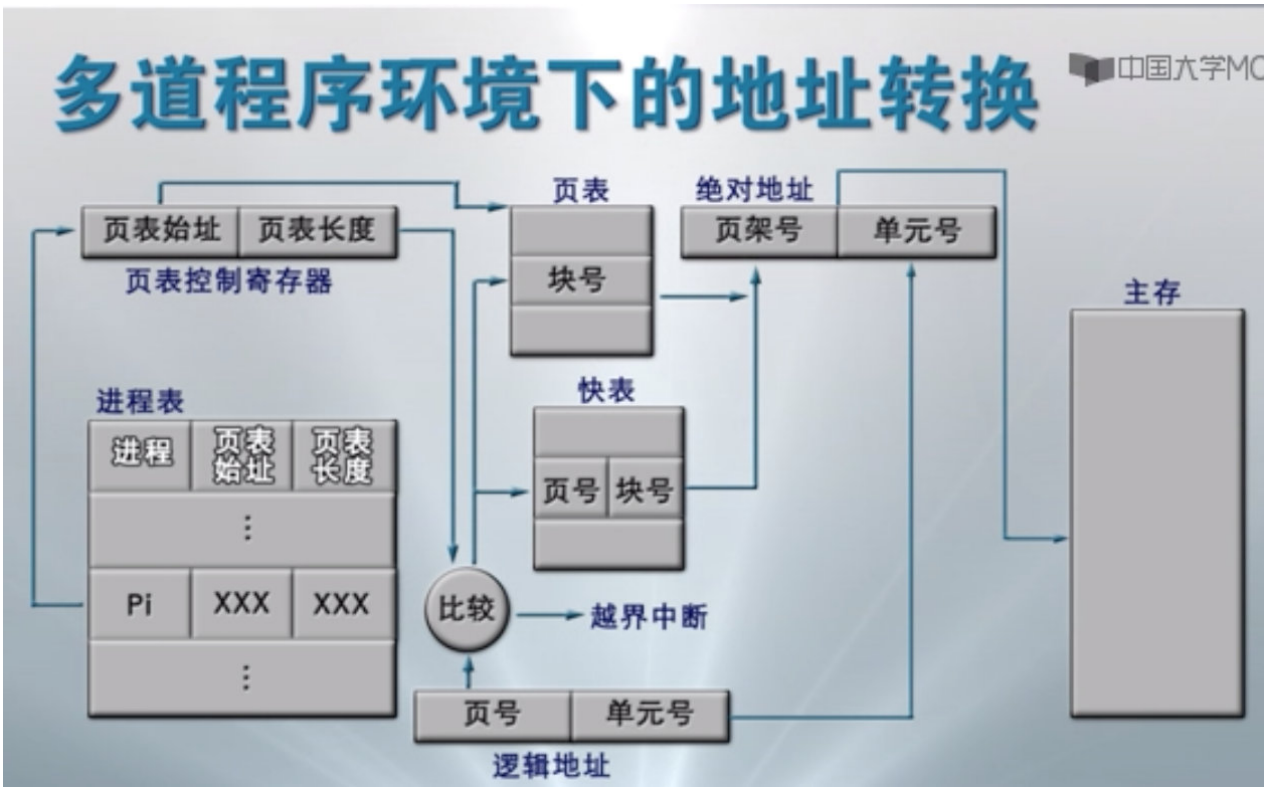
快表：存放在告诉存储区中的页表部分

快表表项：页号+页架号

这种高速存储器是 **联想存储器** ，即按照内容寻址，而非按照地址访问。







2. 段式存储管理

段式 程序设计：把一个程序设计成多个段，用户可以自己应用 **段覆盖技术** 扩充内存空间使用量。这一技术是程序设计技术，不是OS 存储管理的功能。

3. 段页式存储管理

5.虚拟内存

存储管理把进程全部信息放在辅存中，执行时先将其中一部分装入主存，以后根据执行行为随用随调入。

如果主存中没有足够的空闲空间，存储管理需要根据执行行为把主存中暂时不用的信息调出的辅存上去。

实现思路：

- 需要建立与自动管理两个地址空间：
  - （辅存）虚拟地址空间：容纳进程装入
  - （主存）实际地址空间：承载进程执行
- 对于用户，计算机系统具有一个容量大的多的主存空间，即虚拟存储器
- 虚拟存储器是一种地址空间扩展技术，通常意义上对用户编程是透明的，除非用户进行高性能的程序设计。

文件

1. 内存映射文件的原理及实现技术 P326
   - 基本思路是把磁盘访问转变为内存访问
   - 原理是
     - 把进程需要访问的文件映射到其**虚地址空间**中
     - 于是便可通过读写这段虚地址进行文件访问
     - 而磁盘访问转变成内存访问。

I/O

1. I/O 软件的四个层次
   1. 用户IO软件：
      - 库函数
      - SPOOLing软件
   2. 独立于设备的IO软件：基本功能是执行适用于所有设备的常用IO功能，并向用户提供一致性接口
      - 设备命名和设备保护
      - 提供与设备无关的块尺寸
      - 缓冲技术
      - 设备分配和状态跟踪
      - 错误处理和报告

3. IO设备驱动程序：包括与设备密切相关的代码，任务是把用户提交的逻辑IO请求转化为物理IO操作的启动和执行。同时监督设备是否正确执行，**管理数据缓冲区**，进行必要的纠错处理。

   （从独立于设备的软件中 接收并执行IO请求）（设备驱动程序是设备专用的）

   - 设备名转换为端口地址
   - 逻辑记录转化为物理记录
   - 逻辑操作转化为物理操作
4. IO中断处理程序：当设备向CPU提出中断请求时，CPU响应请求并转入中断处理程序
   - 检查设备状态寄存器内容，判断产生中断原因，根据IO操作的完成情况进行相应处理
   - 若数据传输有错，应向上层软件报告设备出错信息，实施重新执行
   - 若正常结束，应唤醒等待传输的进程，使其转换为就绪太
   - 若有等待传输的IO命令，应通知相关软件启动下一个IO请求。

操作	层次
(1)用户进程请求打印一个输出文件。	(1)用户空间输入/输出软件。
(2)将一维磁盘块号转换为三维物理地址(柱面、磁道和扇区)。	(2)设备驱动程序。
(3)获得设备驱动程序的入口地址。	(3)设备独立性软件。
(4)将终端输入的字符转换为ASCII码。	(4)设备独立性软件。
(5)设备驱动进程被唤醒。	(5)中断处理程序。
(6)向设备寄存发器写命令。	(6)设备驱动程序。
(7)检查用户是否有权使用设备	(7)设备独立性软件。
(8)将二进制整数转化成ASCII码以便打印(用户层)。	(8)用户空间软件。
(9)维护一个最近使用块的缓存。	(9)设备独立性软件。
(10)设备缓冲区管理	(10) 操作系统IO软件
(11)设备状态跟踪	(11)设备独立性软件
(12)处理设备IO中发生的异常	(12)
(13) 逻辑地址转为 物理地址	(13) 操作系统IO软件
(14) 检查设备状态寄存器内容	(14) 中断处理程序

2. 为什么要引入缓冲技术，如何实现

   P265

   为了

   - 解决CPU与设备之间速度不匹配的矛盾
   - 协调逻辑记录与物理记录大小不一致的问题

- 提高CPU和设备的并行性
- 减少IO操作对CPU的中断次数
- 放宽对CPU中断响应时间的要走

基本思想：

- 当进程执行写操作输出数据时
  - 先向操作系统申请一个输出缓冲区，然后将数据送至缓冲区
  - 若是顺序写请求，则不断地把数据填入缓冲区，直到装满为止
  - 此后进程可以继续计算
  - 同时，系统将缓冲区的内容写到设备上
- 当进行执行读操作输入数据时，
  - 先向操作系统申请一个输入缓冲区
  - 系统将设备上的一条物理记录读至缓冲区
  - 根据要求把当前所需要的逻辑记录从缓冲区中选出并传送给进程。

附录

简答题

1、操作系统的主要功能包括哪些?

答：操作系统的主要功能包括：处理器管理（处理中断事件、处理器调度）、存储管理（存储分配、存储共享、存储保护、存储扩充）、设备管理、文件管理、作业管理、网络和通信管理。

2、试比较批处理和分时操作系统的不同点?

答：批处理操作系统的主要特征：用户脱机工作、成批处理作业、多道程序运行、作业周转时间长；分时操作系统的主要特征：同时性、独立性、及时性、交互性。

3、进程最基本的状态有哪些？哪些事件可能引起不同状态之间的转换?

答：（1）进程最基本的状态：运行态、就绪态、等待态。

（2）当进程被选中时，就绪态变为运行态；当进程遇到中断时，运行态变为等待态；当等待事件结束时，等待态变为就绪态；当进程即将运行时遇到外部事件的响应，进程由运行态变为就绪态。

4、试说明进程的互斥和同步两个概念之间的区别?

答：进程的互斥和同步两个概念之间的区别：主要是进程对于资源的使用是出于竞争还是协作的关系。

5、什么是临界区和临界资源？对临界区管理的基本原则是什么?

答：（1）临界区：每个进程中访问临界资源的那段程序叫做临界区。进程对临界区的访问必须互斥，每次只允许一个进程进去临界区，其他进程等待。

（2）临界资源：指每次只允许一个进程访问的资源，分硬件临界资源、软件临界资源。

（3）临界区管理的基本原则是：①如果有若干进程要求进入空闲的临界区，一次仅允许一个进程进入。②任何时候，处于临界区内的进程不可多于一个。如已有进程进入自己的临界区，则其它所有试图进入临界区的进程必须等待。③进入临界区的进程要在有限时间内退出，以便其它进程能及时进入自己的临界区。④如果进程不能进入自己的临界区，则应让出CPU，避免进程出现“忙等”现象。

6、试比较分页式存储管理和分段式存储管理?

答：页和分段系统有许多相似之处，但在概念上两者完全不同，主要表现在：

（1）页是信息的物理单位，分页是为实现离散分配方式，以消减内存的外零头，提高内存的利用率；或者说，分页仅仅是由于系统管理的需要，而不是用户的需要。段是信息的逻辑单位，它含有一组其意义相对完整的信息。分段的目的是为了能更好的满足用户的需要。

（2）页的大小固定且由系统确定，把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而一个系统只能有一种大小的页面。段的长度却不固定，决定于用户所编写的程序，通常由编辑程序在对源程序进行编辑时，根据信息的性质来划分。

（3）分页的作业地址空间是维一的，即单一的线性空间，程序员只须利用一个记忆符，即可表示一地址。分段的作业地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址。

7、简述各种I/O控制方式及其主要优缺点?

答：（1）I/O控制方式：询问方式、中断方式、DMA方式、通道方式。

（2）主要优缺点：

程序直接控制方式：由用户来直接控制内存或CPU和外围设备之间的数据传送。

它的优点是：控制简单，也不需要多少硬件支持

它的缺点是：CPU和外围设备只能串行，设备串行，无法发现和处理由于设备或其他硬件所产生的错误。

,中断控制方式：利用向CPU发送中断的方式控制外围设备和CPU之间的数据传送

优点：大大提高了CPU的利用率，支持多道 程序和设备并行。

缺点：占用大量CPU时间，中断次数多，发生中断丢失的现象，数据丢失现象。

DMA方式：在外围设备和内存之间开辟直接的数据交换通路进行数据传送，

优点：在数据传送开始需要CPU的启动指令，结束时发中断通知CPU进行中断处理之外，不需要CPU的干涉。

缺点：在外围设备越来越多的情况下，多个DMA控制器的同时使用，会引起内存地址的冲突并使得控制过程进一步复杂

④通道方式：使用通道来控制内存或CPU和外围设备之间的数据传送，通道是一个独立于CPU的专管I/O的机构，控制内存与设备直接进行数据交换，有自己的通道指令。这些指令受CPU启动，并在操作结束时向CPU发中断信号

优点：减轻CPU的工作负担，增加了并行工作程度

缺点：增加额外的硬件，造价昂贵。

8、⚠️叙述SPOOLING系统和作业调度的关系。

答：SPOOLING系统是**用一类物理设备模拟另一类物理设备** 的技术，是使**独占使用的设备** 变成**多台虚拟设备** 的一种技术，也是一种**速度匹配**技术。作业调度程序根据预定的调度算法选择收容状态的作业运行，作业表是作业调度程序进行作业调度的依据，是SPOOLING系统和作业调度程序共享的数据结构。

9、什么叫“按名存取”？文件系统是如何实现按名存取文件的?

答：（1）“按名存取”是指用户用为方便文件信息的存储和检索标识的信息。

（2）当用户要求存取某个文件时，系统查找目录项并比较文件名就可以查找到所寻文件的目录项，通过目录项指出的文件名可查到所寻文件的目录项，然后通过目录项指出文件信息相对位置或文件信息首块物理位置等就能依次存取文件信息。

《计算机操作系统》名词解释及简答题

- 并发与并行
    - 并发性指两个或两个以上的事件或活动在同一时间间隔内发生;并行性指两个或两个以上的事件或活动在同一时刻发生。并行的事件或活动一定是并发的，但反之并发的事件或活动未必是并行的。并行性是并发性的特例，而并发性是并行性的扩展。
- 分时系统
    - 允许多个联机用户同时使用一个计算机系统进行交互式计算机的操作系统称为分时操作系统
- 实时调度算法
    - 调度那些存在时间上的紧迫性的进程或任务。
- 实时与分时
    - 实时强调在一定时间条件下做出响应，
    - 分时强调同时多用户交互。
- 模式切换，进程切换，两者之间的关系 模式切换时 CPU 从核心态到用户态，或从用户态到核心态 进程切换是指从一个进程上下文切换到另外的进程上下文 模式切换不一定导致进程切换 进程切换一定是先发生模式切换
- 对换与替换与切换
    - 对换指进程粒度的(中级)调度，
    - 替换是存储管理的页面操作，
    - 切换是指进程上下文或模式的改变操作。
- 管道与通道 管道是连接读写进程的一个**特殊文件**，允许进程按 FCFS 方式传送数据，也能够使进程同步执行。 通道又称**I/O 处理机** ，具有自己的指令系统，能完成主存储器和设备之间的信息传送，与 CPU 并行执行的操作。
- 进程 是一个可并发执行的且具有独立功能的程序关于某个数据集合的一次执行过程，也是操作系统进行资源分配和保护的基本单位。
- 线程 线程是进程中能够并发执行的实体，是进程的组成部分，也是处理器调度和分派的基本单位。
- 进程与程序(作业)
    - 进程是程序关于数据的执行。程序是静态的，进程是动态的。
- 进程通信及其分类
    - 进程之间相互交换信息的工作称之为进程通信，可以通过高级通信机制来完成。进程间通信的方式包括:
        - 通过软中断提供信号的通信机制;
        - 使用信号量及其原语操作(PV、读写 锁、管程)控制的**共享存储区**通信机制;
        - 通过**管道**提供的共享文件通信机制;
        - 使用信箱和发信/收信原语的**消息传递**通信机制。
- 线程的实现机制有哪几种，试比较各种实现机制的优缺点

    - 内核级实现 KLT
        - 优:在多处理器上，能同时调度统一进程中的多个线程并行执行;切换速度快，提高系统效率
    - 缺:系统开销大用户级实现
    - ULT
        - 优:节省模式切换开销和内核宝贵资源;按需选择调度算法;能运行在任何操作系统上
        - 缺:一个用户级线程的阻塞将引起整个进程的阻塞;不 混合实现 优:宏观和微观上都有很好的并行性;系统开销小，窗口系统执行效率高
- 管程及其特性
    - 由表示共享资源的数据结构及其上的一组操作过程组成，是一种程序设计语言结构成份，和信号量有同等的表达能力。主要特性:共享性、安全性、互斥性。
- 死锁
    - 如果在一个进程集合中，每个进程都在等待只能由该集合中的其他一个进程才能引发的事件，则称一组进程或系统此时发生了死锁。
- 死锁与饥饿
    - 死锁:一组进程如果都获得部分资源，还想要得到其它进程所占有的资源，最终所有进程将陷入永远等待状态
    - 饥饿:一个可运行进程由于其它进程总是优先于它，而被调度程序无限期地拖延而不能被执行。
- 进程死锁的必要条件及几种死锁处理方法
    - 互斥条件，占有和等待条件，不剥夺条件，循环等待条件 死锁的避免、死锁的防止、死锁检测与解除
- 文件
    - 由文件名字标识的一组信息的集合
- 文件存取方式
    - 顺序、直接、索引
- 内存映射文件和优点
    - 内存映射文件技术把进程的虚拟地址空间与某一个盘文件关联起来，使得进程对文件的存取转化为对关联存储区域的访问，通过文件系统与存储管理相结合来实现。
    - 优点:方便易用、节省空间、便于共享、灵活高效的特点。

- 文件目录与目录文件 为了加快文件的查找速度，通常把 FCB(文件控制块)集中起来进行管理，组成文件目录。目录项的格式按统一标准定义，全部由目录项所构成的文件称为目录文件。
- 文件的共享方式
- 动态、静态、符号链接
- ⚠️操作系统
    - 操作系统是管理系统资源、控制程序执行、改善人机界面、提供各种服务、合理组织计算机工作流程、为用户有效使用计算机提供良好运行环境的系统软件。主要特性:并发性、 共享性、异步性。
- 操作系统的用户接口
    - 操作系统提供两种用户接口:操作(命令)接口和程序(系统调用)接口

- 驱动调度
    - 系统运行时，同时会有多个访问辅助存储器的进程请求输入、输出操作，操作系统必须采用一种调度策略，使其能按最佳(最有效)的次序执行各访问请求。
- I/O 控制方式(计算机输入/输出控制方式)、各种控制方法内容及特点1、
    - 程序直接控制方式:又称轮询方式，使用查询指令测试设备控制器的忙碌状态位，确定 主存储器和设备是够能交换数据。耗费大量的 CPU 时间、无法检测设备错误、只能串行工作。
    - 2、中断控制方式:数据放设备缓冲区，缓冲区满了就发生中断，把数据给 CPU。并行操作的设备数受到中断处理时间的限制。CPU 仍 需花较多的时间处理中断。中断次数增多时易导致数据丢失。

  3、直接内存存取方式 DMA:不经过 CPU，DMA 和主存储器之间直接字传送。 简单，价格低廉;“周期窃取”降低CPU 处理效率;功能不够强，不能满足复杂的 I/O 请求 4、通道方式:给 CPU 发出 I/O 启动命令后，由通道指令完成启动设备等工作。提高整个系统效率;大大减少 CPU 和设备之间的逻辑联系
- 处理器调度层次及主要内容
    - 高级调度:又称作业调度、长程调度，在多道批处理操作系统中，从输入系统的一批作业中按照预定的调度策略挑选若干作业进入主存，为其分配所需资源，并创建作业的相应用 户进程。
    - 中级调度:又称平衡调度、中程调度，根据主存资源决定主存中所能容纳的进程数目， 并根据进程的当前状态来决定辅助存储器和主存中的进程的对换。
    - 低级调度:又称进程调度/线程调度、短程调度，根据某种原则决定就绪队列中的哪个 进程/内核级线程获得处理器，并将处理器出让给它使用。
- 中断 在程序执行过程中，遇到急需处理的时间时，暂时中止现行程序在 CPU 上的运行，转而执行相应的事件处理程序，待处理完成后再返回断点或调度其他程序执行。
- 硬中断与软中断
    - 通过硬件设施来产生请求，称作硬中断。
    - 利用硬件中断的概念，用软件方式进行模拟，实现宏观上的异步执行效果的中断称作软中断。
- 中断及异常
    - 中断是由与现行指令无关的中断信号触发的(异步的)，且中断的发生与 CPU 处在用户模式或内核模式无关，在两条机器指令之间才可响应中断，一般来说，中断处理程序提供的服 务不是为当前进程所需的，中断与 CPU 的异步的
    - 异常是由处理器正在执行现行指令而引起的，一条指令执行期间允许响应异常，异常处 理程序提供的服务是为当前进程所用的。异常包括很多方面，有出错(fault)，也有陷入(trap)等。异常与 CPU 是同步的
- MMU主存管理部件
    - 提供地址转换和存储保护能力，并支持虚拟存储器和多任务管理。
- MMU（存储管理部件）： P217
    - 操作系统的存储管理依靠底层硬件支撑来完成任务，此硬件称为存储管理部件
    - 它提供**地址转换**和**存储保护**功能并支持**虚存管理**和**多任务管理**。
- 内核
- 内核是提供支持系统运行的基本功能和基本操作的一组程序的模块，分为微内核和单内核。
- PSW程序状态字，
    - 用于区别不同的处理器工作状态(处于何种状态，能否执行特权指令),

    - 主要作用是方便地实现程序状态的保护和恢复。
- 临界区
    - 并发进程中与共享变量有关的程序段称为“临界区”。
    - 共享变量所代表的资源称为“临界资源”。
- 快表
- 为了提高运算速度，在硬件中设置相联存储器，用来存放进程最近访问的部分页表项，称作快表。
- 设备独立性 申请设备时，用户指定逻辑设备，使得用户作业和物理设备分离考来，再通过其他途径建立逻辑设备和物理设备之间的映射，设备的这种特性称为“设备独立性”。
- 影响缺页中断率的主要因素
    - 主存页框数、页面大小、页面替换算法、程序特性
- 集中分布资源管理与完全分布资源管理
    - 集中：对所管资源拥有完全的控制权，一类资源中的每一个资源仅受控于一个资源管理者
        - 投标算法
        - 由近及远算法
        - 回声算法
    - 完全：对所管资源仅有部分控制权，不仅一类资源存在多个管理者，且该类中每个资源都由多个管理者共同管理。
        - Lamport 算法
        - RICart 算法
        - Bull算法
        - 令牌环算法
- 虚拟存储器
    - 在具有层次结构存储器的计算机系统中，自动实现部分装入和部分替换的功能，能从逻辑上为用户提供一个比物理主存容量大得多的、可寻址的“主存储器”。
- 虚拟机
    - 一台抽象计算机，它在硬件基础上由软件来实现，并且与物理计算机一样，具有指令集及可用的存储空间。
- 安全性主要内容包括:安全策略、安全模型、安全机制(认证、授权、加密、审计)
    - 安全策略
        - 访问支持策略
        - 访问控制策略
    - 安全模型
        -
    - 安全机制
        - 认证：认证机制用于阻止非法用户侵入系统，措施包括标识、鉴别和身份认证
        - 授权：授权机制是拒绝或防止已进入系统的进程执行违反安全策略的任何操作的设施。
            - 授权
            - 确定访问权限
            - 试试存取权限
        - 加密：加密是用某种方式**伪装信息**以隐藏其内容的过程

        - 审计： 审计又称安全审计，是对系统中有关安全活动进行**完整记录**、**检查及审核**，作为一种事后追踪手段来保证系统的安全性，是对系统安全性所实施的一种技术措施，也是对付计算机犯罪者的主要利器。

- DAC 和 MAC(自主访问控制和强制访问控制)
    - DAC 是资源主属可以按照自己的意愿制定系统中其他用户对其资源的访问控制权限的一类访问约束机制
    - MAC 用于将系统中的信息分密级和范畴进行管理，保证每个用户只能 够访问那些被标明能够由他访问的信息的一种访问约束机制。
    - （强制访问控制：安全系统通过比较主、客体的响应标记来决定是否授权一个主体对客体的访问权限）
- MAC 比 DAC 有更强的安全手 段和设施，使用户不能通过意外事件和有意的误操作逃避安全控制.
- 试从资源管理的角度，分析操作系统的作用和功能。
    - 对资源进行抽象研究，找出各种资源共性和个性，有序地管理计算机中的硬件、软件资源，跟踪资源使用情况，监视资源的状态，满足用户对资源的需求，协调各程序对资源的使 用冲突;
    - 研究使用资源的统一方法，让用户简单、有效的使用资源，最大限度地实现各类资 源的共享，提高资源利用率，从而，使得计算机系统的效率有很大提高。
- 简述进程之间的关系有哪几种，并分析典型的有界环形缓冲器生产者-消费者问题中生产者 消费者进程之间的关系
    - 协作和竞争
    - 生产者--消费者问题是计算机操作系统中并发进程内在关系的一种抽象，是典型的进程 同步问题。Pi 和 Ci 都是并发进程，只要缓冲区未满，生产者进程 Pi 所生产的产品就可以投 入缓冲区;只要缓冲区非空，消费者进程 Ci 就可以从缓冲区取走并消耗产品。
- 根据信号量和 P、V 操作的定义可以得到哪些推论，请简要叙述。
    - 推论 1 若信号量 s.value 为正值，此值等于在封锁进程之前对信号量 s 可施行的 P 操作数，亦即 s 所代表的实际可用的物理资源
    - 推论2 若信号量s.value为负值，其绝对值等于登记排列在s信号队列之中等待的进程个数，即恰好等于对信号量 s 实施 P 操作而被封锁并进入信号量 s 等待队列的进程数。
    - 推论 3 P 操作通常意味着请求一个资源，V 操作意味着释放一个资源，在一定条件下，P 操作代表挂起进程的操作，而 V 操作代表唤醒被挂起进程的操作
- 简述操作系统的几个主要功能，以及现在操作系统的主要特征?
    - 功能:处理机管理、存储管理、设备管理、文件管理、网络与通信管理、用户接口
    - 特性:并发性、共享性、异步性
- 来自处理器和主存内部的中断称“异常”，列举它的分类及主要区别? 答:
- 异常处理程序提供的服务是为当前进程所用的。异常包括出错和陷入。
- 出错和陷入的主要区别是:
    - 它们发生时保存的返回指令地址不同，出错保存指向触发异常的那条指令，而陷 入保存指向触发异常的那条指令的下一条指令。因此，当从异常返回时，出错会重新执行那 条指令，而陷入就不会重新执行那条指令。如缺页异常是一种出错，而陷入主要应用在调试 中。
- 叙述 LRU 页面置换算法的思想，并给出 3 种可能的实现方案。
    - 答:根据程序局部性原理，那些刚被使用过的页面，可能马上还要被使用，而在较长时间里 未被使用的页面，可能不会马上使用到。LRU 算法淘汰的页面是在最近一段时间里较久未被 访问的那页。

    - ⚠️可能的实现方案:
        - 页面淘汰队列、
        - 标志位法、（引用位法）
        - 多位寄存器法（老化算法）
        - 多位计数器法等。（计数法）
- 在一个分布式系统中，如何对系统中的事件进行一致性排序?
    - 答:对分布式系统中的每个结点来说，事件的排序由下列规则确定:对于来自站点 i 的消息x 和来自站点 j 的消息 y，若下列条件之一成立，则说事件 x 先发生于事件 y，如果: (1)Ti < Tj 或 (2)如果Ti=Tj 并且 i<j
- 试解释多级页表与反置页表。
    - 答:多级页表:在大地址空间的情况下，为了节省页表内存占用空间，可设计成两级(或多 级)页表，即页表也分成一张张页表页(大小等于页面)，并不全部放入内存，虚地址分成三部 分:页目录表、页表页、位移，通过页目录索引找页表页，通过页表页索引找到对应页框号， 并与位移一起形成物理地址。反置页表:反置页表为内存中的物理块建立一个页表并按照块 号排序，该表的每个表项包含正在访问该页框的进程标识、页号及特征位，和哈希链指针等， 用来完成内存页框到访问进程的页号，即物理地址到逻辑地址的对应转换。
- ⚠️叙述 SPOOLING 系统的技术特点、组成和数据结构。
    - 答:spooling 系统是能把一个物理设备虚拟化成多个虚拟(逻辑)设备的技术，能用共享设备 来模拟独享设备的技术，在中断和通道硬件的支撑下，操作系统采用多道程序设计技术，合 理分配和调度各种资源，实现联机的外围设备同时操作。
    - spooling 系统主要有:预输入、井管理和缓输出组成，
    - 数据结构包括:作业表、预输入表和缓输出表。
- 解释操作系统体系结构分类，说明各种结构的主要特点。
    - 答:操作系统体系结构分类有整体式结构、层次式结构、虚拟机结构、客户服务器及微内核 结构等。整体式结构高效但不便维护修改，层次式结构便于维护但效率低，虚拟机结构方便 资源管理使用，客户服务器及微内核结构便于扩充但通信开销大。
- 试比较虚拟存储管理与中级调度中对换技术的区别
    - 虚拟存储管理:以页或段为单位处理
    - 进程所需主存容量大于当前系统空闲量时仍能运行 对换技术(中级调度，挂起和解除挂起):以进程为单位处理进程所需主存容量大于当前系统空闲量时，无法解除挂起

- ⚠️试比较分页式存储管理与分段式存储管理。
    - 分段，是信息的逻辑单位，由源程序的逻辑结构所决定，用户可见，段长由用户确定， 段起始地址可以从任何主存地址开始
    - 分页，是信息的物理单位，与源程序的逻辑结构无关，用户不可见，页长由系统确定， 页面只能以页大小的整倍数地址开始。分页机制 将进程信息的副本存放在辅助存储器中，当他被调度投入运行时，并不把程序和数据全部装入主存，仅装入当前使用的页面，进程执行过程中访问到不在主存的页面时，再把所需 信息 动态的装入。
- 作业调度和低级调度算法
    - 先来先服务算法(FCFS):非剥夺式调度算法
    - 最短作业优先算法(SJF):非剥夺式调度算法，作业所要求的 CPU 时间最短的作业优先
    - 最短剩余时间优先算法(SRTF):剥夺式调度算法，剩余运行时间最少优先
    - 响应比最高者优先算法(HRRF):非剥夺式调度算法，响应比等于 1+作业等待时间/作业处 理时

间，最大优先
    - 优先级调度算法:非剥夺式和剥夺式
    - 轮转调度算法(RR):剥夺式调度
- 可变分区搜索分配算法
    - 最先(first fit)适应分配算法:顺序查找未分配区表或链表，直到找到第一个能满足长度要 求的空闲区为止，分割此分区，一部分分配给作业，另一部分仍为空闲区。
    - 下次(next fit)适应分配算法:从未分配区的上次扫描结束处顺粗查找未分配区或链表，直 到找到第一个能满足长度要求的空闲区为止，分割此分区，一部分分配给作业，另一部分仍 为空闲区。
    - 最优(best fit)适应分配算法:扫描整个未分配区或链表，从空闲区中挑选一个能满足客户 进程要求的最小分区进行分配。
    - 最坏(worst fit)适应分配算法:扫描整个未分配区或链表，总是挑选一个最大的空闲区分 割给作业使用。
- 页面替换算法
    - 最佳页面替换算法(OPT):淘汰以后不再访问的页，或距现在最长时间后要访问的页面。
    - 先进先出页面替换算法(FIFO):淘汰主存中驻留时间最长的页面。 最近最少使用页面替换算法(LRU):淘汰的页面是在最近一段时间内最久未被访问的那一 页。
    - 最近未使用页面替换算法(NRU):引用位 R
    - 第二次机会页面替换算法(SCR):最先进入主存的页面如果最近还在被使用，仍然有机会像 新调入页面一样留在主存中。
    - 时钟页面替换算法(Clock)
    - ✨局部最佳页面替换算法(MIN):滑动窗口 t，在(a, t+a)里没有将要使用的页面就 out工作集(WS):滑动窗口 t，在(a-t, a)里没有将要使用的页面就 out
- 搜查定位(磁盘请求)
    - “先来先服务”算法(FCFS):磁盘臂随机移动。
    - “电梯调度”算法:每次总是选择沿移动方向最近的那个柱面，若同一柱面上有多个请求， 还需进行旋转优化。
    - “最短查找时间优先”算法:先执行查找时间最短的请求。
    - “扫描”算法:移动臂每次沿一个方向移动，扫过所有柱面，遇到最近的 I/O 请求便进行处 理，直到到达最后一个柱面后，再向相反的方向移动回来。
    - “循环扫描”算法:上述扫描算法中，到达最后一个柱面后，不是反方向，而是同一方向从 另一端开始扫描。
- .可通过哪些途径来提高内存利用率?
    - 答：内存利用率不高，主要有四种表现形式：1）内存存在着大量的、分散的难以利用的碎片；2）暂时不用或长期不能运行的程序或数据，占据了大量的存储空间；3）当作业较大时，内存中只能装入少量的作业，当其阻塞时，将使CPU空闲，从而降低了内存利用率；4）内存中存在着重复的拷贝。
    - 针对上述问题，可采用以下方法提高内存利用率：1）改连续分配为离散分配；2）增加对换机制；3）引入动态链接机制；4）引入虚拟存储器机制；5）引入存储器共享机制。
- 设备管理中的数据传送控制方式有哪几种？分别简述如何实现的。
    - 答：
    - 程序直接控制：由用户进程来直接控制内存或CPU和外设间的信息传送。
    - 中断方式：进程通过CPU发出指令启动外设，该进程阻塞。当输入完成时，I/O控制器通过中断请求线向CPU发出中断信号，CPU进行中断处理。
    - DMA方式：在外设和内存之间开辟直接的数据交换通路。

    - 通道控制方式：CPU发出启动指令，指出通道相应的操作和I/O 设备，该指令就可启动通道并使该通道从内存中调出相应的通道指令执行。
- 说明作业调度，中级调度和进程调度的区别，并分析下述问题应由哪一级调度程序负责。

  (1) 在可获得处理机时，应将它分给哪个就绪进程；

  (2) 在短期繁重负载下，应将哪个进程暂时挂起。

  答：

    - 作业调度用于决定把外存中处于后备队列中的哪些作业调入内存，并为它们创建进程，分配资源，然后将新创建进程插入就绪队列；
    - 中级调度负责将内存中暂时不具备运行条件的进程换到外存交换区存放，但内存空闲时，又将外存中具备运行条件的进程重新换入内存；
    - 进程调度决定将处理机分配给就绪进程队列的哪个进程。（4分）
    - (2)进程调度、中级调度
- 计算虚存访问时间
    - 有快表
        - 快表命中： 访问快表+ 访问主存
        - 快表不中& 页表命中：访问快表 + 访问页表（主存）+ 访问主存
        - 快表不中& 页表不中： 访问快表+ 访问页表+ 访问辅存（传送页面）+ 访问页表（主存）+ 访问主存
    - 无快表
        - 页表命中：访问页表（主存）+ 访问主存
        - 页表不中：访问页表+ 访问辅存（传送页面）+ 访问页表（主存）+ 访问主存
- 文件共享方式 P323
    - 静态共享：两个或多个进程可以通过**文件链接**共享一个文件，无论进程是否运行，文件链接关系都是存在的。
    - 动态共享：指系统中不同的应用进程或同一用户的不同进程并发地访问同一文件，这种共享关系只有当进程存在是才能出现。
    - 符号链接共享：符号链接（软链接）是只有文件名，不指向inode的链接，通过名称来引用文件。
- ⚠️画出 SPOOLing 结构图，工作原理?
    - SPOOLing技术是用一类物理设备模拟另一类物理设备的技术，是使独占型设备变成共享型设备的一种技术。
- ⚠️系统调用：P24
    - 系统调用把应用程序的请求传送至内核，调用相应**服务例程**完成所需处理，将处理结果返回给应用程序。
    - (操作系统提供程序使用的系统服务函数或过程)
- 底半处理：
    - 为缩短中断处理的屏蔽时间，提高系统的并发工作能力，而采用的一种任务延迟处理机制，核心代码在关中断的核心态完成与中断事件有关的基本处理，而另一部分耗时的工作留在中断处理例程之外，在开中断的非核心态完成。
- 常见的实时系统调度算法是
    - 单比率调度
    - 期限调度
    - 最少裕度调度

- 操作系统的主要特征是
    - 并发性
    - 共享性
    - 异步性
    - 虚拟性
- 操作系统：是管理系统资源，控制程序执行，改善人机界面，提供各种服务，合理组织计算机工作流程和为用户有效使用计算机提供良好的运行环境的一种系统软件。
- 文件d 存储空间管理方法：
    - 位视图
    - 空闲区表
    - 空闲块链
    - 空闲块列表
- 文件
    - 逻辑：流式（无结构）、记录式
    - 物理：顺序、链接、索引
- 一个实时系统使用了4个周期事件，其周期分别为50ms，100ms，200ms，250ms 。假设这4个周期事件分别需要35ms，20ms，10ms和xms的CPU时间，保持系统可调度的最大x值是多少
    - 对于任务1. 一秒内最多可以执行 1000/50 次， 极端情况是，这一秒内都有任务在占用CPU。
    - 1s中，4个事件分别需要的CPU时间为： 1000/50×35=700ms 1000/100×20=200ms 1000/200×10=50ms 1000/250×x=4xms 因此，700+200+50+4x=1000，可以得到 x=12.5ms。
- 有一台计算机，具有1MB内存，操作系统占用200KB，每个用户进程各占200KB。如果用户进程等待I/O的时间为80%，若增加1MB内存，则CPU的利用率提高多少?
    - **答：**设每个进程等待I/O的百分比为P，则n个进程同时等待I/O的概率是Pn ，当n个进程同时等待I/O期间CPU是空闲的，故CPU的利用率为1-Pn 。由题意可知，除去操作系统，内存还能容纳4个用户进程，由于每个用户进程等待I/O的时间为80%，故：

      CPU利用率=1-(80%)4 =0.59

      若再增加1MB内存，系统中可同时运行9个用户进程，此时：

      CPU利用率=1-(80%)9 =0.87

      故增加1MB内存使CPU的利用率提高了47%：

      87%÷59%=147%

      147%-100%=47%