# 导论

操作系统的功能作用：1、作用：操作系统是控制和管理计算机系统内各种硬件和软件资源，有效地组织多道程序运行的系统软件，使用户与计算机之间的接口。2、功能：处理机管理、存储管理、设备管理、文件管理、用户接口。

操作系统的发展过程：1、手工操作时期：人工干涉，用户独占。2、早期批处理时期：出现了完成作业自动转换工作的程序叫监督程序，包括早期联机批处理、早期脱机批处理。3、多道批处理系统：在内存中同时存放多道程序在管理程序的控制下交替执行，用户独占。分时系统：用户与主机交互。实时系统：具有专用性及时性。4、现代操作系统（网络操作系统和分布式操作系统）：有网络地址，提供网络服务，实现资源共享。

# 第一章 计算机系统结构

计算机系统在硬件方面的保护：**1、双重模式操作**：用户模式、监督程序模式，双重模式操作为人们提供了保护操作系统和用户程序不受错误用户程序影响的手段。 **2、I/O 保护**：定义所有 I/O 指令为特权指令，所以用户不能直接发出 I/O 指令，必须通过操作系统来进行 **3、内存保护**：对中断向量和中断服务程序进行保护，使用基址寄存器和界限寄存器 **4、CPU 保护**：使用定时器防止用户程序运行时间过长，操作系统在将控制权交给用户之前，应确保设置好定时器，以便产生中断。

# 第二章 操作系统结构

操作系统的基本组成：进程管理、内存管理、文件管理、输入/输出系统管理、二级存储管理、联网、保护系统、命令解释系统。

系统调用的含义：系统调用提供了进程与操作系统之间的接口。分为五类：进程控制、文件管理、设备管理、信息维护、通信。

操作系统设计所采用的结构：**1、简单结构**：以较小、简单且功能有限的系统形式启动，但后来渐渐超过了其原来的范围，由于运行所用的硬件有限，它被编写成利用最小的空间提供最多的功能 **2、分层方法**：优点：模块化，简化了调试和系统验证；缺点：涉及对层的仔细认真的定义的困难，效率较差 **3、微内核**：优点：便与操作系统扩充，便于移植；缺点：关于哪些服务应保留在内核内，而哪些服务应在用户空间内实现，并没有定论。

# 第三章 进程

进程的含义：进程是由数据结构以及在其上执行的程序组成，是程序在这个数据集合上的运行过程，也是操作系统进行资源分配和保护的基本单位。进程包含的内容：程序代码、程序计数器、堆栈段、数据段。进程和程序的联系和区别：1、程序是静态的，进程是动态的，程序是代码的集合，进程是程序的执行。2、程序是永久的，进程是短暂的。3、程序的组成是代码，进程的组成包括程序、数据、进程控制块。3、一个程序可对应多个进程，通过调用关系，一个进程也可包括多个程序。4、进程可生成其他进程，程序不能生成新的程序

并发与并行的区别：并行指两个或多个事件在同一时刻发生；并发指两个或多个事件在同一时间间隔内发生。在单处理机系统中，每一时刻仅能执行一个程序，所以从微观角度来看程序的并发执行时，它们是交替执行，即串行。程序间真正的并行只有在多处理机系统中才会出现。

进程的状态及转换条件：状态：新的，运行，等待，就绪，终止。转换条件：**1、新建--〉就绪**：就绪队列能够容纳新的进程 **2、就绪—〉运行**：处于就绪状态的进程被调度程序选中，分配到 CPU。**3、运行—〉阻塞**：正在运行的进程因某种条件为满足而放弃对 CPU 的占用。**4、阻塞—〉就绪**：

处于阻塞状态的进程所等待的事情发生了。**5、运行—〉就绪：**正在运行的进程用完了本次分配给它的 CPU 时间 **6、运行—〉终止：**正在运行的进程完成自己的工作或者由于发生某些事件而被异常中止。
进程控制块的作用及内容：（1）内容：进程状态、程序计数器、CPU 寄存器、CPU 调度信息、内存管理信息、记账信息、I/O 状态信息。（２）作用：①PCB 是进程组成中最关键的部分，每个进程都有惟一的 PCB,操作系统对每个进程实施控制和管理，②系统创建一个进程时，为它创建一个 PCB，当进程终止时，系统收回其 PCB,该进程在系统中就不存在了，所以 **PCB 是进程存在的惟一标志**。
上下文切换的过程：将 CPU 切换到另一个进程需要保存原来进程的状态并装入新进程的保存状态。过程如下：① 保存处理器内容② 对当前运行进程的 PCB 进行更新③ 将这个进程的 PCB 移入适当的队列④ 挑选其他进程执行⑤ 对挑选进程的 PCB 进行更新⑥ 对存储器管理数据结构进行更新⑦ 将被选中进程上次移出时的处理器状态进行恢复。
进程间通信的机制有哪些，包括单机系统和网络系统：1、单机系统：进程通信根据交换信息量的多少分为高级通信和低级通信。低级通信方式：P，V 操作。高级通信方式：共享存储器方式、消息传递方式、管道文件。网络系统：套接字、远程过程调用、远程方法调用。

# 第 4 章　线程

一、定义：是 CPU 使用的基本单元；他由现程ＩＤ，程序计数器，寄存器集合和堆栈组成。
二、优点：１、响应度高 2、资源共享 3、经济 4、多处理器体系结构的利用
三、多线程模型
１、多对一模型：允许开发人员随意创建任意多的用户线程，但是由于内核只能一次调度一个线程，所以并不能增加并发性。
２、一对一模型：提供更大的并发性，但是创建内核的开销会影响应用程序的性能。
３、多对对模型：没有以上两者缺点。

# 第 5 章　CPU 调度

一．　调度的层次
1. 长程调度：又称为作业调度或高级调度。长程调度决定哪些作业可参与竞争 CPU 和其他资源。一般功能是：从后备作业中挑选一个或多个作业，为它(它们)分配基本的内存和外设资源并建立相应的进程。
2．中程调度：也称中级调度。中级调度决定哪些进程可参与竞争 CPU，用以实现进程的活动状态与静止的挂起态之间的转换。引入中程调度的主要目的，是为了提高内存的利用率和系统吞吐量。中程调度实际上就是存储器管理中的置换功能。
3．短程调度：又称为进程调度或低级调度。短程调度决定哪个进程可获得物理 CPU。短程调度的功能是：从就绪进程队列中挑选一个进程并启动执行该进程。短程调度是最基本的一种调度，它可以采用非抢占方式或抢占方式
二．　调度的方式
可抢占：提高响应性，进来后马上使用 CPU。不可抢占：一旦把 CPU 分配给一个进程，那么该进程就会保持 CPU 直到终止或轮转到等待状态。
三．　调度算法
**１．先来先服务（ＦＣＦＳ）　　非抢占式的。**
FCFS 策略遵循先来者优先服务的原则，
例如：进程 Burst 时间：P1　24秒，　P2　3秒，P3　3秒
如果进程以P1、P2、P3 的顺序到达，并且以FCFS 规则服务，我们将获得如下的甘特图：

| P1 | P2 | P3 |
|---|---|---|

0 24 27 30

P1 的等待时间是0 毫秒，进程P2 是24 毫秒，P3 是27 毫秒。这样，平均时间是(0 + 24 + 27)/3 = 17毫秒。

优点：它简单易行

缺点：调度性能较差，有可能使短的进程长期等待长的进程（护送效应）。它有利于偏重 CPU 的进程，不利于偏重 I/O 的进程。

**２．短作业优先调度算法（ＳＪＦ） 抢占式的／非抢占式的**

当CPU 有效时，它将被赋给下一个CPU burst 最小的进程。

例如：进程 Burst 时间 P１ 6秒；P2 8秒；P3 7秒；P4 3秒

用SJF 调度，我们将依照如下的甘特图来调度这些进程：

| P 4 | P１ | P３ | P２ |
|---|---|---|---|

0 3 9 １６ ２４

P1 的等待时间是3 毫秒，P2 是16 毫秒，P3 是9 毫秒，P4 是0 毫秒。因而，平均等待时间是(3 + 16 + 9 +0)/4 = 7 毫秒。如果使用FCFS 调度策略，那么平均等待时间是（０＋６＋１４＋２１）／４＝10.25 毫秒。

新进程的下一个CPU burst 可能会比当前执行进程的剩余量短。在抢占式SJF 算法中，新进程抢占当前的进程；而非抢占式SJF 算法允许当前运行的进程结束其CPU burst.

例如：

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 8 |
| P2 | 1 | 4 |
| P3 | 2 | 9 |
| P4 | 3 | 5 |

按照抢占式SJF 调度会产生如下的结果：

| P1 | P2 | P4 | P1 | P3 |
|---|---|---|---|---|

0 1 5 10 17 26

P2 在时间1 到达。进程P1 的剩余时间大于进程P2的时间需求（4 毫秒），所以进程P1 被抢占，P2 被调度。这个例子的平均等待时间是((10 - 1) + (1 - 1) + (17- 2) + (5 - 3))/4 = 26/4 = 6.5 毫秒（被调度时间减去到达的时间）。而采用非抢占式SJF 调度的平均等待时间为（（０－０）＋（８－１）＋（１２－３）＋（１７－２））／４＝7.75 毫秒（p1, p2, p4, p3）.

优点：可证明SJF 调度算法是最佳的算法，因为它为指定的进程组给出了最小的平均等待时间。对短作业或短进程最为有利。

缺点：对长作业不利，特别是在抢占方式下，可能会使长作业无限延迟。而且实际上无法预测下一CPU 瞬时段的长度。

**3. 优先级调度算法 抢占式的／非抢占式的**

为每个进程赋予一个优先权，把CPU 分配给拥有最高优先权的进程。

缺点：优先调度算法可能导致一些低优先权进程无限的等待CPU（即无限阻塞）

解决一个方法是老化（aging））是指逐渐地提高在系统中长时间等待的进程的优先权.

**4. 轮转调度算法（ＲＲ） 抢占式的**

ＣＰＵ分配给每一个进程一段时间，到时间轮转到下一个进程

若 需要ＣＰＵ的时间<分配的时间量 自己释放ＣＰＵ，等待下一次轮转

若 需要ＣＰＵ的时间>分配的时间量 进程一结束就释放ＣＰＵ，让给下一个进程使用。

**５．多级队列调度算法**

就绪队列划分为多个独立的队列。每个队列有自己的调度算法。进程不会在队列间移动。

在队列间进行调度，这种调度通常实现为权限固定的抢占式调度或采用时间片机制。

**６．多级反馈队列调度算法 抢占式的**

它综合考虑了多种因素，根据进程运行情况的反馈信息动态改变进程的优先级，按优先级别组织设置了规定 CPU 时间片的各就绪队列；对同一队列中的各就绪进程按 FCFS 或 RR 算法调度。
优点：这种策略不必事先知道各种进程所需的执行时间，而且还可以满足各种类型进程的需要。如果一个 CPU 占用了太多的 CPU 时间，它将被移动到优先权更低的队列（lower-priority queue）。这种策略把 I/O 繁忙型进程和交互式进程放置到优先权更高的队列（higher-priority queue）。类似的，一个在低优先权队列中等待太久的进程可能会被移动到一个更高优先权的队列。这种老化形式避免了饥饿的发生。
例如； 进入就绪队列的进程被放到队列0。队列0 中的进程被指定一个8 毫秒的时间量。如果它不能在这个时间内完成，将被移动到队列1 的队尾。如果队列0 是空的，那么队列1 头部的进程被指定一个16 毫秒的时间量。如果它不能完成，将被抢占并被放入队列2。队列2 中的进程只有当队列0 和1 为空时以FCFS原则运行。
四．调度准则
１．ＣＰＵ利用率（越大越好） ２．吞吐量（越大越好）３．周转时间：从进程提交到进程完成的时间间隔（越小越好）４．反应时间：从进程提交请求到首次产生响应的时间（越小越好）５．等待时间：在就绪队列中等待的时间（越小越好）

# 6. 进程同步

一． 进程为什么需要同步和互斥
进程的同步：一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态
进程的互斥 ：由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程间竞争使用这些资源，进程的这种关系为进程的互斥
二 .信号量的含义
信号量表示资源，初始化值非负。信号量 s ＞＝ 0 时， s 表示可用资源的数量。执行一次 wait 操作意味着请求分配一个单位资源，因此 s 的值减 1 ；当 s ＜ 0 时，表示已经没有可用资源。请求者必须等待别的进程释放该资源，它才能运行。而执行一个 siginal 操作意味着释放一个单位资源，因此 s 加 1 。若 s ＜＝ 0 ，表示某些进程正在等待资源。因此要唤醒一个等待状态的进程，使之运行下去。
临界资源： 系统中某些资源一次只允许一个进程使用，称这样的资源为临界资源或互斥资源或共享变量
临界区（互斥区）：在进程中涉及到临界资源的程序段叫临界区。
三．信号量和自悬锁的区别
自悬锁：当一个进程处于临界区时，其它进程持续循环等待。繁忙等待浪费了CPU 周期，本来其它进程可以有效利用这些时间。在多处理机系统中非常有用。自旋锁的优点是：当期望短时间持有锁时，自旋锁很有用。（小王一直等着别人出来，自己不去睡觉）
信号量：为了克服对繁忙等待的需求，我们修改wait 和signal 信号量操作的定义。当一个进程执行wait 操作且信号量不为正时，它必须等待。然而，进程可以阻塞自身，而不是繁忙等待。block 操作将一个进程置入与该信号量关联的等待队列中并将进程状态设为等待。（小王可以去睡觉，等别人唤醒）
四．经典的同步互斥问题
１． 生产者消费者问题
（１）一个生产者，多个消费者，公用一个缓冲区
定义两个同步信号量
Empty——表示缓冲区空否 初始值为 1
Full——表示缓冲区满否 初始值为 0
生产者进程

While(True )
{
生产一个产品；
P(empty);
产品送往 buffer;
V(full); //唤醒消费者
}
消费者进程
While(True )
{
P(full);//看是否有产品
从 buffer 中取一个产品；
V(empty); //唤醒生产者
消费该产品
}

（２）一个生产者，多个消费者，公用 n 个缓冲区

定义两个同步信号量

Empty——表示缓冲区空否　　初始值为 n

Full——表示缓冲区满否　　初始值为 0

生产者进程 i=0

While(True )
{
生产一个产品；
P(empty);
产品送往 buffer[i];
V(full);
i=(i+1)%n
}
消费者进程 j=0
While(True )
{
P(full);
从 buffer[j]中取一个产品；
V(empty);
消费该产品
j=(j+1)%n
}

（３）m 个生产者，k 个消费者，公用 n 个缓冲区

这个问题不仅生产者与消费者要同步，而且各个生产者，消费者之间还要互斥的访问缓冲区

定义三个信号量

Empty——表示缓冲区空否　　初始值为 n

Full——表示缓冲区满否　　初始值为 0

Mutex——协调进程间的互斥　　初始值为 1

生产者进程
producer(i)
While(True )
{

```
    生产一个产品；
    P(empty);
    P(mutex);
      产品送往 buffer[i];
      i=(i+1)%n
    V(mutex);
      V(full);
  }
  消费者进程
    consumer(i)
  While(True )
  {
    P(full);
    P(mutex);
      从 buffer[j]中取一个产品；
      j=(j+1)%n
    V(mutex);
      V(empty);
      消费
  }
```

注意：i=(i+1)%n 不能放在 V(full)之后 。否则会导致第一个生产者放完后没有加 1，第二个生产者又放入同一个缓冲区。（一个缓冲区只能放一个产品）

P(full)不能和 P(mutex)颠倒位置，否则会导致死锁。考虑一种极端情况，前 n 个生产者生产了 n 个产品，占满了 n 个缓冲区。此时，empty=0,mutex=1。当第 n ＋ 1 个生产者 mutex=0,empty=-1,生产者进程阻塞，切换到消费者进程，mutex=－ 1 消费者进程阻塞，第 1 ~k 个消费者等待 V(mutex)释放，第 n+1~m 个生产者等待 V(empty)释放，造成死锁

2． 桌面上有一空盘子，允许存放一个水果。爸爸可以向盘中放苹果，也可以放橘子，儿子专等吃盘中苹果，女儿专等吃橘子。规定盘中空时，一次只能放一只水果吃者食用。实现三人并发进程的同步

三个信号量：S1 盘子是否为空 初始值 1
S2 盘子是否有苹果 初始值为 0
S3 盘子是否有橘子 初始值为 0

```
  父亲
  P(S1)
  放
 if(apple)
            V(S2)
  else
             V(S3)
儿子
P(S2)
  取苹果
V(S1)
  吃苹果

女儿
P(S3)
```

取橘子
V(S1)
吃橘子

哲学家就餐问题
5 个信号量 chopstick[5]表示 5 根筷子　　初值均为 1
1 个信号量 room　　初值为 4 （只能由 4 个人进去吃饭）

```
do {   p(room)
p(chopstick[i])
p(chopstick[(i+1) %5])
eat;
v(chopstick[i])
v(chopstick[(i+1) %5])
v(room)
think;
```

5．　理发店问题
见课后习题及老师网上给出的课后答案
应注意的问题
1．信号量表示资源
P(s)取资源
V（s）释放资源
2．P(s) ，V（s）要配对。
若这对出现在两个进程中　　解决同步
若这对出现在同一个进程中　　解决互斥
３．　同步与互斥同时存在时，同步在互斥

# 第 7 章：死锁

1、产生死锁的四个必要条件：互斥、占有并等待、非抢占、循环等待。
2、死锁的处理方法：第一个是可使用协议以预防或避免死锁，确保系统决不会进入死锁状态。第二个是允许系统进入死锁状态，然后检测它，并加以恢复。第三个是可忽略这个问题，认为死锁不可能在系统内发生。
为确保死锁不会发生：系统可采用死锁预防或死锁避免方案
死锁预防：是一组方法，以确保至少一个必要条件不成立
死锁避免：要求操作系统事先得到有关进程申请资源和使用资源的额外信息。
3、死锁预防的处理方法
互斥：不能解决，原因：对于非共享资源，必需要有互斥条件
占有并等待：保证觉一个进程申请一个资源时，他不能占有其他资源。
非抢占 L 如果一个进程占有资源并申请另一个不能立即分配的资源，那么其现已分配都被抢占。即这些资源都被隐式的释放了。
循环等待：对所有的资源编号类型进行完全排序，且要求每个进程按递增顺序来申请资源。
4、死锁避免
安全状态：系统能在一定的资源上合理的分配资源给进程，以完成进程的执行，从而这种状态称为安全状态。
安全序列：设进程顺序<p1,p2,……,pn>，如果对于每个 Pi，Pi 申请的资源小于当前可用资源加上所有进程 Pj（其中 j<i）所占有的资源，那么这个顺序为安全序列。
不安全状态：如果没有这样的顺序存在，那么系统就要处于不安全状态。
安全状态不是死锁状态，相反，死锁状态是不安全状态。
5、银行家算法：(见翻译版书 188 页)。算法包括安全性算法和资源请求算法。
6、死锁检测：资源分配图→等待图的转换 （有环，即死锁）
7、死锁恢复：第一：进程终止。第二：资源抢占（包括选择一个牺牲品、回滚、饥饿）。

练习题：见书上 197 页的 8-13.

# 第 8 章　　内存管理

一．　内存管理的基本功能有哪些？

在多道程序系统中，要求存储管理具有内存空间管理（空间的分配和回收）、地址转换、内存扩充、内存保护和共享，小作业大内存等功能。

二．逻辑地址和物理地址的区别

CPU 产生的地址通常被称为逻辑地址。而物理地址则是内存单元的地址。

三．连续内存分配的方法和缺点

其基本原理是给每一个内存中的进程划分一块适当大小的存储块，以连续存储各进程的程序和数据，使各进程能并发进行。

1. 固定分区法

固定分区法就是把内存固定划分为若干个不等的区域，划分的原则由系统决定。在整个执行过程中保持分区长度和分区个数不变。（每个程序找适合自己的区）

2. 动态分区法

动态分区分配是根据进程的实际需要，动态地为它分配连续的内存空间，各个分区是在相应作业装入内存时建立的，其大小恰好等于作业的大小。为了实现分区分配，系统中设置了相应的数据结构来记录内存的使用情况。

缺点：

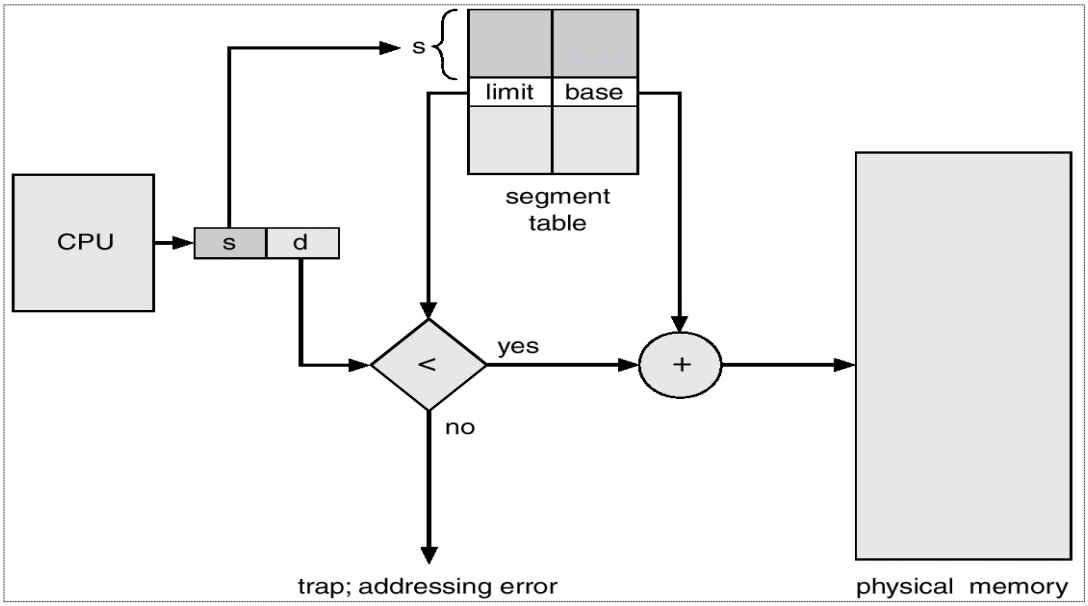
４．　动态重定位分区分配算法

与动态分区分配算法基本上相同；差别仅在于：在这种分配算法中，增加了“紧凑”功能，通常是在找不到足够大的空闲分区来满足用户需求时，进行紧凑处理。（解决碎片问题）

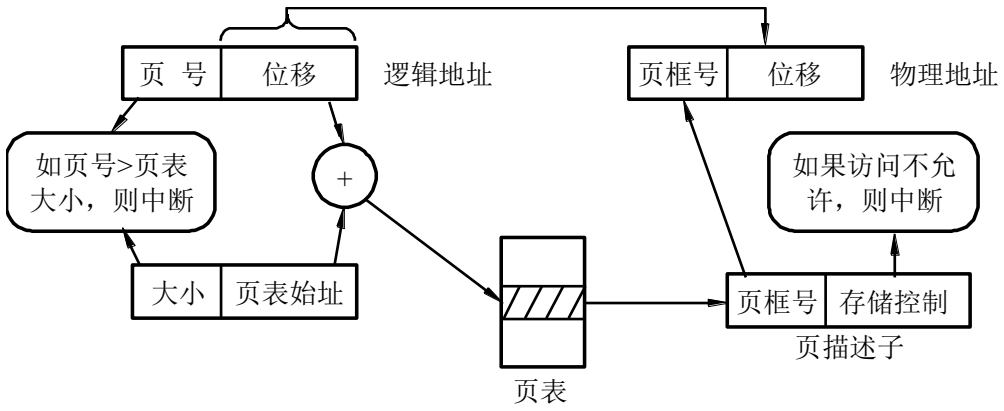
缺点：浪费严重，内存利用率低，固定内存分区产生内部碎片。动态分区产生外部碎片

四．书后练习题（老师上课留的）

**分页机制**：分页管理也是解决碎片问题的一种有效办法，它允许程序的存储空间是不连续的。




1.页号：将一个进程的逻辑地址空间划分成若干个大小相等的部分，每一部分称为页，并且每页都有一个编号。
帧号：将内存空间也划分成与页面大小相同的若干个存储块，即为物理块
2.页表的作用是实现从页号到物理块号的地址映射。
3.快表是为了加快地址转换速度而使用的一个联想高速缓存（不使用快表，要访问两次内存）
4.内存管理单元 MMU 主要是用来完成地址转换
它具有如下功能：将逻辑地址分为页号和页内位移，先查快表，若没有，在页表中找到与该页对应的物理块号，再将其与页内位移一起组成物理地址；当页表中的状态位为“0”，或页面访问越界，或出现保护性错误时，内存管理单元会出现页面失效异常。（见课件 char9 P27）
或参看下图：




优点：分页机制不会产生外部碎片：任何空闲帧都被分配给所需的进程　　缺点：是它分离了内存的用户视角和实际的物理内存。用户所见内存与实际的物理内存并不相同。
**分段机制：**
分段方法是将程序分成若干逻辑段，并对这些段分别分配存储空间。这些段的长度可以不同，也不必连续进行分配。

分段式存储管理的优点：

- 提供了内外存统一管理的虚存，每次调入一段有意义的信息。
- 允许段长动态增长。

- 便于对具有完整逻辑功能的信息段进行共享。
- 便于实现动态链接。

分段式存储管理的缺点：需要更多的硬件支持，诸多功能会使系统的复杂性大大增加；段的长度受内存可用区大小的限制；若替换算法选择不恰当就有可能产生抖动现象。

五．外部碎片和内部碎片
内部碎片：
块内多余的内存就是内部碎片
分页会产生内部碎片，不会产生外部碎片
外部碎片：
分段会产生外部碎片不会产生内部碎片
六．页面置换算法
1．先进先出页面置换算法（FIFO）
方法是将最先进入队列的页号所对应的页面最先选择为牺牲者。这种方法易于理解，但性能不是在任何场合都是好的。　　使用 FIFO 方法可能会出现 Belady 异态，这是一种在增加帧的情况下反而使缺页中断率增加的异常情况
2．最佳页面置换算法
这种方法总是替换最长将来时间不被使用的那个页面。它需要访问将来知识。通常用来同其他方法进行比较。
3．最近最少使用页面置换算法
根据页面调入内存后的使用情况，选择最近最少使用的页面予以淘汰。即选择距本页最远的页，将其替换。
4．第 2 次机会页面置换算法
每个页有一个访问位。先选择“最老”的页，若其访问位被清除（为 0），则它就是牺牲者；若它的访问位已置值（为 1），则先清除它（置 0），然后选择下一页，重复前述过程

# 第 9 章 虚拟内存

虚拟内存时将用户逻辑内存与物理内存分开。通常请求页面调度来实现代码。
请求页面调度处理过程：
当换入进程时，调页程序推测在该进程两次换出之前会用到哪些页。
调页程序不是患如整个进程，而是把那些必须也调入内存。
1、检查进程的页表，以确定该引用是合法还是非法的地址访问。
2、如果引用非法，那么终止进程。如果引用有效但尚未调入页面，那末现在应调入。
3、找到一个空闲帧
4、调度一个磁盘操作，已将所需要的页调入刚分配的帧。
5、当磁盘读操作完成后，修改进程的内部表和页表，以表示该页已在内存中。
6、重新开始因非法地址陷阱而且中断的指令。
页置换算法
FIFO 也值换　（置一页时，选择最久的一页）Belady 异常
最优页置换　（置换最长时间中 不会被使用的页）
LRU 页置换 （置换最长时间没有使用的页）
LRU 近似页置换：附加引用位算法 二次机会算法（增强型二次机会算法）
系统颠簸
如果一个进程在该页上用的时间要多于执行时间，那这个进程就在颠簸。
原因：1 频繁的忙于页的调进和调出。
　　2　分配的帧数少于现有局部的大小

## 第 10 章　文件系统接口（老师没讲，但要自己看，故略）

## 第 11 章　文件系统实现

I/O 控制为最底层，由设备驱动程序和终端处理程序组成，实现内存和磁盘之间的信息转移。
应用程序——逻辑文件系统——问题组织模块——基本文件系统——I/O 控制——设备
目录实现方法：线性列表和哈希表
磁盘的分配方法：连续分配，链接分配，索引分配
空闲空间管理：位向星，链表

# 第 12，13 章

字符设备：是按一个字节一个字节的传输
块设备：以块为单元进行传输
网络设备：以套接字为单位进行传输的
I/O 调度：调度一组 I/O 请求就是确定一个好的顺序来执行这些请求
缓冲：缓冲区是用来包存两设备之间火灾设备和应用程序之间所传输数据的内存区域。（三个理由①处理数据流的生产者与消费者之间的速度差异②协调传输数据大小不一致的设备③应用程序 I/O 的拷贝语义。
高速缓存：可以保留数据拷贝的高速内存.
假脱机和设备预留：
Spool 是用来包存设备输出的缓冲（打印机的例子）
错误处理：
作为一个规则，I/O 系统调用返回一位调用状态信息，以表示成功或失败。
操作系统简述题
1：操作系统的组成部分：
（１）进程管理；（２）作业管理；（３）存储管理；
（４）输入输出管理；（５）Ｉ／Ｏ软件（６）文件系统
２：硬件保护机制：
（１）I/O Protection；（２）Memory Protection；（３）CPU Protection；
（４）Hardware Address Protection；（５）Hardware Protection
３：系统调用的概念
　系统调用提供了进程和操作系统间的接口，具体包括：
（１）进程控制；（２）文件管理；（３）设备管理；（４）信息维护；（５）通信
4：操作系统设计的基本结构有哪些, 优缺点
（1）串行处理系统：
（2）简单批处理系统
（3）多道程序批处理系统 ：通过将用户的 CPU 请求和 I/O 请求重叠起来的办法来有效地使用 CPU。它设法让 CPU 总有事情可做，以此来提高 CPU 的利用率
（4）分时系统：当程序正在执行时，允许用户与计算机交互，从而大大缩短了系统的响应时间，增加了用户调试程序和控制程序运行的灵活性
（5）实时系统：处理事务必须在适合于此系统的特定时间限额内完成
5：多级反馈队列调度的优点
多级反馈队列调度算法允许进程在队列间移动。其思想是利用 CPU burst 特性区分进程。如果一个 CPU 占用了太多的 CPU 时间，它将被移动到优先权更低的队列（lower-priority queue）。这种策略

把 I/O 繁忙型进程和交互式进程放置到优先权更高的队列（higher-priority queue）。类似的，一个在低优先权队列中等待太久的进程可能会被移动到一个更高优先权的队列。这种老化形式避免了饥饿的发生。

6：系统颠簸的原因和解决方法

７：文件系统对空闲空间的管理方法

（1） 连续分配：就是将文件存放在辅存的连续存储区中；

（2） 索引分配：利用文件分配表ＦＡＴ给每个文件分配一个指出该文件索引表所在物理块号的表目；

（3） 链接分配：就是将文件存放在非连续的存储区中；

８：硬盘空间分配方法及各自优缺点

９：目录实现的方式

（1） 单级目录：文件系统在每个存储设备上仅建立一个目录文件的目录结构；

（2） 二级目录：二级目录结构将存储在设备上的目录文件分成两级。第一级为系统目录（主目录），包含用户目录名和指向该用户目录的指针；第二级为用户目录，包含用户所有文件的文件目录；

（3） 多级目录；

１０：分段与分页的各自优缺点

（１）对于纯分页系统而言，它有效地解决了内存碎片问题，有可能让更多的作业同时投入运行，提高了处理机和存储器的利用率；缺点是：由于要求运行的作业必须一次全部装入内存，当作业要求的空间大于当前可用存储空间时，作业只有等待这使作业地址空间受到内存实际容量的限制，并且要对每个作业建立和管理相应得页表，还要增加硬件实现地址转换，增加了系统时间和空间上的开销；

对于请求式分页系统而言，消除了内存碎片，同样增加了系统时间和空间上的开销，但由于每个作业部分装入就可以投入运行，大大增加了作业的利用空间，提高了内存的利用率，使作业地址空间不再受内存容量大小的限制，缺点是缺页时要进行页面交换，会引起系统“抖动”

（２）分段是一种内存管理机制，它支持内存的用户视角提供了内外存统一管理的虚存， 每次调入的是一段有意义的信息；

分段机制允许段长动态增长

分段机制便于对具有完整逻辑功能的信息段进行共享；

分段机制便于实现动态链接；

缺点是：比其他方式需要更多的硬件支持，并且诸多功能的实现是的系统的复杂性大大增加，另外段的长度也受到内存可用区大小的限制；若替换算法选择不恰当就可能产生“抖动”现象

１１：I/O 子系统完成的管理功能

（１）设备命名；（2）设备保护；（3）与设备无关的块及存储设备的块分配；

（４）设备分配；（５）出错处理