一、单项选择题（本大题共 12 小题，每小题 1 分，共 12 分）

二、多项选择题（本大题共 14 小题，每小题 2 分，共 28 分）

三、填空题（本大题共 10 空，每空 1 分，共 10 分）

四、判断分析题（本大题共 5 小题，每小题 4 分，共 20 分）

五、分析计算题（本大题共 2 小题，每小题 6 分，共 12 分）

六、综合运用题（本大题共 2 小题，每小题 9 分，共 18 分）

*第五章和第七章（虚拟存储、文件管理）题目分值共 6 分（选择题）

第一章

画重点

操作系统基本概念

解释：是一组能有效组织和管理计算机硬件和软件资源，合理对各类作业进行调度，以及方便用户使用的程序的集合。

目标：方便性、有效性、可扩充性、开放性

有效性：提高系统资源的利用率、提高系统吞吐量

三个作用：

1、作为用户与计算机硬件系统之间的接口

2、作为计算机系统资源的管理者

3、实现了对计算机资源的抽象

四个特征：

1、并发：两个或多个事件在同一时间间隔内发生

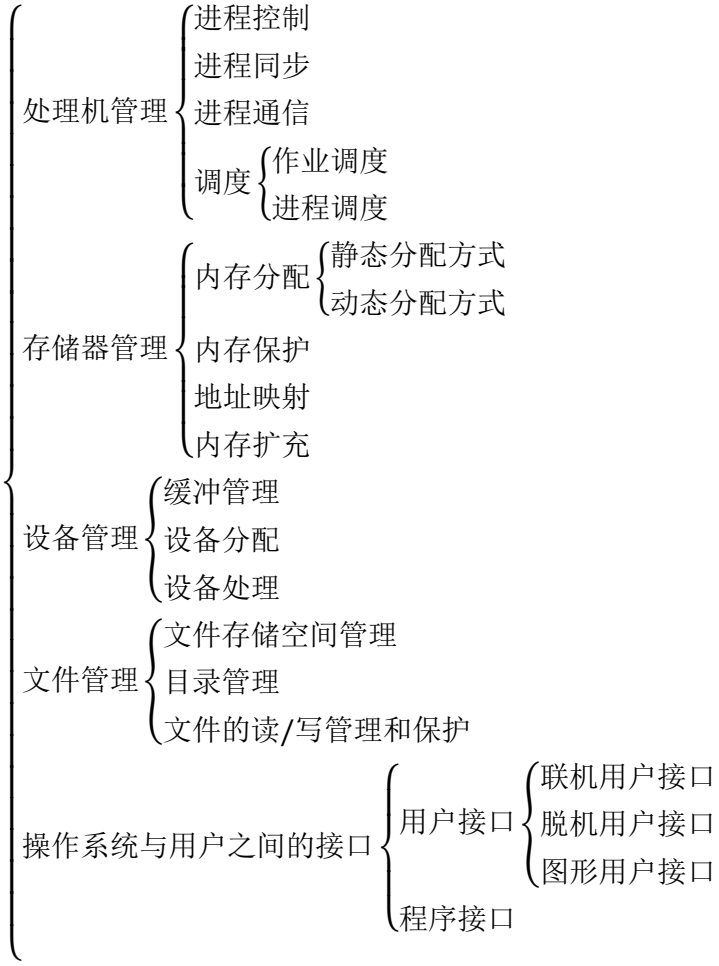
2、共享：通过互斥共享方式和同时访问方式实现

3、虚拟：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称为“虚拟”。在OS中通过时分复用技术和空分复用技术实现。

4、异步：进程以人们不可预知的速度向前推进。

性能指标：并发数、吞吐率、响应时间

主要功能：翻页有个花花括号



好多好多操作系统

	解释	优点	缺点
人工操作方式	将事先已穿孔的纸带/卡片装入纸带输入机/卡片输入机，再启动它们，将纸带/卡片上的程序和数据输入计算机，启动计算机运行。程序运行完毕并取走计算结果后，允许下一个用户上机		1、用户独占全机 2、CPU等待人工操作
脱机输入输出方式	事先将装有用户程序和数据的纸带装入纸带输入机，在一台外围机的控制下，把纸带/卡片上的数据/程序输入到磁带上。当CPU需要这些程序和数据时，再从磁带上高速调入内	1、减少CPU的空闲时间：各种操作在脱机情况下由外围机完成 2、提高I/O速度：CPU运行中需要输	

	存。	入数据时。直接从高速的磁带上将数据输入到内存	
单道批处理系统	step1.由监督程序将磁带上的第一个作业装入内存，把运行控制权交给该作业 step2.作业处理完成，控制权交还给监督程序 step3.监督程序把磁带上的第二个作业调入内存		系统中的资源得不到充分利用
多道批处理系统	由作业调度程序按一定的算法，从后备队列中选择若干个作业调入内存，使它们共享CPU和系统中的各种资源	1、资源利用率高 2、系统吞吐量大 3、平均周转时间长	无交互能力
分时系统	在一台主机上链接了多个配有显示器和键盘的终端，并由此所组成的系统，该系统允许多个用户同时通过自己的终端，以交互方式使用计算机，共享主机资源	**多路性**：多台终端可以连接到一台主机上，按分时原则为用户服务。 **独立性**：每个用户在各自终端上操作，互不干扰 **及时性**：用户的请求能在很短的时间内获得响应 **交互性**：用户可通过终端与系统进行广泛的人机对话	
实时系统	系统能及时响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。进一步分为： 工业（武器）控制系统 信息查询系统 多媒体系统 嵌入式系统	**主要特征**：将时间作为关键参数 **多路性**：（信息查询系统的多路性与分时系统一样）系统周期性地对多路现场信息进行采集，对多个对象或多个执行机构进行控制 **独立性**：对信息的采集和对对象的控制彼此互不干扰 **及时性**：以控制对象所要求的截止时间确定 **交互性**{信息查询系统： 多媒体系统： **可靠性**：要求系统高度可靠，往往采取多级容错措施来保障系统的安全性及数据的	

	安全性

微机操作系统

	解释	主要配置	优秀代表
单用户单任务操作系统	只允许一个用户上机，且只允许用户程序作为一个任务运行。	8位和16位微机	CP/M MS-DOS
单用户多任务操作系统	允许一个用户上机，但允许用户把程序分为若干个任务，使他们并发执行	32位微机	windows
多用户多任务操作系统	允许多个用户通过各自的终端，使用同一台机器，共享主机系统中的各种资源。 每个用户程序又可进一步分为几个任务，使他们能并发执行，进一步提高资源利用率和系统吞吐量。	大中小型机 32位微机	UNIX {Solaris OS Linux OS

多道程序设计优点

（1） 提高CPU的利用率

（2） 提高内存和I/O设备利用率

（3）增加系统吞吐量

作业

（ D ）操作系统是一种 **I** ，在操作系统中采用多道程序设计方式能提高 CPU 和外部设备的 **II** 。一般来说，为了实现多道程序设计，计算机需要有 **III** 。

	(1)	(2)	(3)	(4)
I	通用软件	系统软件	应用软件	软件包
II	利用效率	可靠性	稳定性	兼容性
III	更大的内存	更快的外部设备	更快的 CPU	更先进的终端

A.133

B.213

C.131

D.211

为了提高计算机的处理机和外部设备的利用率，把多个程序同时放入主存，在宏观上并行运行是⑧；

把一个程序划分成若干个同时执行的程序模块的设计方法是⑨；

多个用户在终端设备上的交互方式输入、排错和控制其程序的运行是①；

由多个计算机组成的一个系统，这些计算机之间可以通信来交换信息，互相之间无主次之分，它们共享系统资源，程序由系统中的全部或部分计算机协同执行，管理上述计算机系统的操作系统是⑤；

有一类操作系统的系统响应时间的重要性超过系统资源的利用率，它被广泛地应用于卫星控制、飞机订票业务等领域是②。

①分时 OS

②实时 OS

③批处理系统

④网络 OS

⑤分布式 OS

⑥单用户 OS

⑦多重程序设计

⑧多道程序设计

⑨并发程序设计

（单选）操作系统的不确定性是指（ D ）

A.程序运行结果的不确定性

B.程序运行次序的不确定性

C.程序多次运行时间的不确定性

D.B&C

（单选）在操作系统中，负责进程调度的是

A.处理机管理

B.信息管理

C.存储管理

D.设备管理

（单选）不属于操作系统性能指标的是

A.作业的大小

B.资源利用率

C.吞吐量

D.周转时间

（判断）在主存容量为 M 的多用户分时系统中，当注册用户数为 N 个时，每个用户拥有的主存空间为 M/N。（ ）

（单选）如果分时系统过的时间片一定，那么（ ），响应时间越长

A.用户数越少

B.内存越多

C.内存越少

D.用户数越多

（单选）（A）是操作系统提供给应用程序的接口。

A.系统调用

B.中断

C.库函数

D.原语

（单选）在操作系统中，并发性是指若干个事件（）发生。

A.在同一时刻

B.一定在不同时刻

C.在某一时间间隔内

D.依次在不同时间间隔内

第二章

画重点

进程

解释：进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。

5个特征：

1、结构特征：为使程序(含数据)能独立运行，应为之配置一进程控制块，即PCB；而由程序段、相关的数据段和PCB三部分便构成了进程实体。

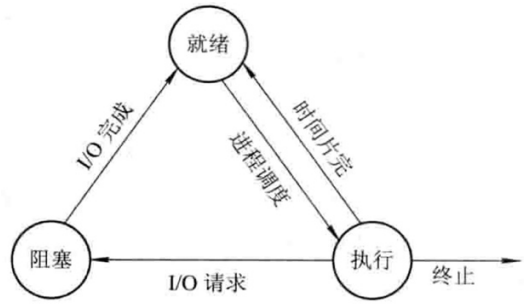
2、动态性：由创建产生，由调度执行，由撤销消亡。

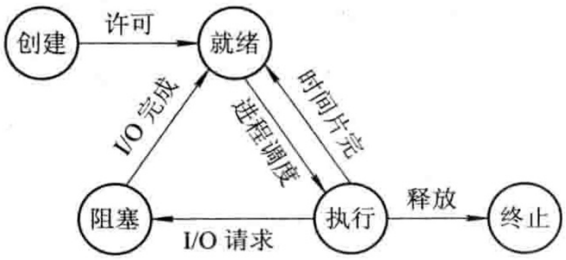
3、并发性：多个进程实体共存于内存中，且能在一段时间内运行，这也是OS的重要特征。

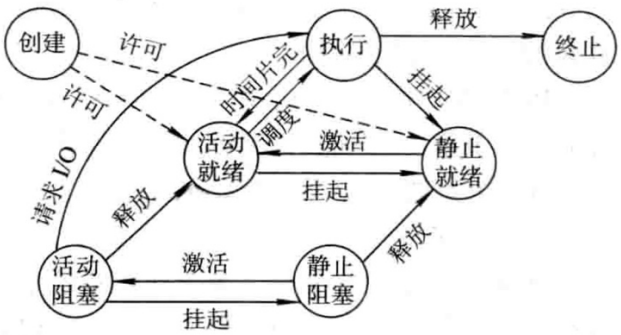
4、独立性：进程实体是一个能独立运行、独立获得资源和独立接受调度的基本单位。未建立PCB的程序都不能作为一个独立的单位参与运行

5、异步性：进程按各自独立的、不可预知的速度向前推进。

进程状态及状态转换







PCB

解释：为了使参与并发执行的每个程序（含数据）都能独立地运行，在操作系统中必须为之配置一个专门的数据结构，称为进程控制块。进程利用PCB来描述进程的基本情况和活动过程，进而控制和管理进程。

作用：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

进程同步

概念：对多个相关进程在执行次序上进行协调，使并发执行的诸进程之间按照一定的规则或时序共享系统资源，并能很好的相互合作，从而使程序的执行具有可再现性。

两种制约关系$\begin{cases}\text{间接相互制约}\\ \text{直接相互制约}\end{cases}$

遵守四条准则：

1、空闲让进：无进程处于临界区时，表明临界资源处于空闲状态，应允许一个请求进入临界区的进程立即进入自己的临界区，以有效地利用临界资源。

2、忙则等待：当已有进程进入临界区时，表明临界资源正在被访问，因而其它试图进入临界区的进程必须等待，以保证对临界资源的互斥访问。

3、有限等待：对要求访问临界资源的进程，应保证在有限时间内能进入自己的临界区，以免陷入“死等”状态。

4、让权等待：进程不能进入自己的临界区时，应立即释放处理机，以免陷入忙等状态。

临界资源

解释：许多硬件资源如打印机、磁带机等，都属于临界资源，诸进程间应采取互斥的方式实现对这种资源的共享

pv 操作

是标准的原子操作，执行时不可中断

p 操作=wait()

```
wait(S)
{
    while(S<=0);
    S--;
}
```

v 操作=signal()

```
signal(S)
{
    S++;
}
```

线程的基本概念

在操作系统中再引入线程，则是为了减少程序在并发执行时所付出的时空开销，使OS具有更好的并发性。

为使多个程序更好地并发执行同时又尽量减少系统的开销，将进程的上述两个属性分开，由操作系统分开处理，亦即对于作为调度和分派的基本单位，不同时作为拥有资源的单位，以做到“轻装上阵”；而对于拥有资源的基本单位，又不对之进行频繁的切换。正是在这种思想的指导下，形成了线程的概念

线程的属性

轻型实体：线程中的实体基本上不拥有系统资源，只是有一点必不可少的、 能保证其独立运行的资源。

独立调度和分派的基本单位：在多线程OS中，线程是能独立运行的基本单位，线程的切换非常迅速且开销小。

可并发执行：在一个进程中的多个线程之间能并发执行、不同进程中的线程也能并发执行。

共享进程资源：在同一进程中的各个线程都可以共享该进程所拥有的资源

生产者-消费者问题

有一群生产者进程在生产产品，将产品提供给消费者进程去消费。为使生产者进程与消费者进程能并发执行，在两者之间设置了一个具有n个缓冲区的缓冲池。

生产者进程将产品放入缓冲区中，消费者进程从一个缓冲区中取走产品。

既不允许消费者到空缓冲区去取产品，也不允许生产者向装满产品的缓冲区中投放产品。

用记录型信号量解决

```
int in = 0, out = 0;//输入指针，输出指针
item buffer[n];//有 n 个缓冲区的缓冲池
semaphore mutex = 1, empty = n, full = 0; //互斥信号量（任何时候只有一个进程使用缓
                                          冲区）、可供使用缓冲区的数量、放有产品的
                                          缓冲区数
```

```
void proceducer()
{
   //生产者
   do
   {
      proceducer an item nextp; ... //生产下一个项目
      wait(empty);
      wait(mutex);
      buffer[in] = nextp;
      in= (in + 1) % n;
      signal(mutex);
      signal(full);
   } while (true);
}
void consumer()
{
   do
   {
      wait(full);
      wait(mutex);
      nextc = buffer[out];
      out = (out + 1) % n;
      signal(mutex);
      signal(empty);
      consumer the item in nextc;
   } while (true);
}
void main()
{
   cobegin
   proceducer();
   consumer();
   coend
}
```

用and信号量解决

```
int in = 0, out = 0;
   item buffer[n];                    //项目缓冲区
   semaphore mutex = 1, empty = n, full = 0;
   void proceducer()
   {
      do
      {
         proceducer an item nextp;//生产下一个项目
         Swait(empty, mutex);
         buffer[in] = nextp;
         in:= (in + 1) % n;
         Ssignal(mutex,full);
      } while (true);
   }
   void consumer()
   {
      do
      {
         Swait(full, mutex);
         nextc = buffer[out];
         out = (out + 1) % n;
         Ssignal(mutex,empty);
         consumer the item in nextc;
      } while (true);
   }
```

哲学家进餐问题

5个哲学家共用一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五只筷子。一个哲学家进行思考，饥饿时取左右最近的筷子，只有拿到两只筷子才能进食。

用记录型信号量解决

semaphore chopsticks[5]={1,1,1,1,1}

第 i 位哲学家的活动可描述为：

do

```
{
    wait(chopstick[i]);
    wait(chopstick[(i + 1) % 5]);
    // ...eat...
    signal(chopstick[i]);
    signal(chopstick[(i + 1) % 5]);
    // ...think...
} while (true);
```

当 5 个哲学家同时拿起左边的筷子时产生死锁，解决办法：

（1）至多只允许四位哲学家同时拿左边筷子

（2）仅当哲学家的左右筷子均可用时，才允许进餐

（3）规定奇数号哲学家先拿左边筷子，偶数号先拿右边筷子

用and型信号量解决

semaphore chopsticks[5]={1,1,1,1,1}

第 i 位哲学家的活动可描述为：

```
do
{
    Sswait(chopstick[(i + 1) % 5], chopstick[i]);
    // ...eat...
    Ssignal(chopstick[(i + 1) % 5], chopstick[i]);
    // ...think...
} while (true);
```

作业

1、什么是前趋图？为什么要引入前趋图？

前驱图是一个有向无环图，用于描述进程之间执行的先后顺序。图中的结点表示一个进程或程序段，有向边表示两个结点之间存在的偏序或前驱关系。

为了更好描述进程地顺序和并发执行情况，所以引入前趋图。

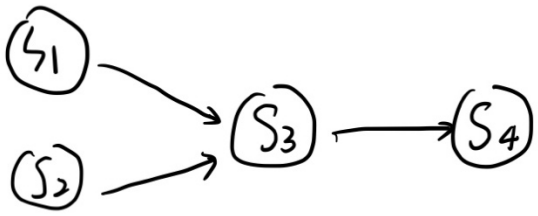
2、试画出下面四条语句的前趋图

S1:a=x+y

S2:b=z+1

S3:c=a-b

S4:w=c+1



4、程序并发执行时为什么会失去封闭性和可再现性？

失去封闭性：当系统中存在着多个可以并发执行的程序时，系统中的各种资源将为它们所共享，而这些资源的状态也由这些程序来改变，致使其中任一程序在运行时，其环境都必然会受到其它程序的影响。例如，当处理机已被分配给某个进程运行时，其它程序必须等待。

不可再现性：并发执行时，由于失去了封闭性，也将导致其又失去可再现性。例如，有两个循环程序A和B，它们共享一个变量N。程序A每执行一次时，都要做N=N+1操作；程序B每执行一次时，都要执行Print(N)操作，然后再将N置成“0”。程序A和B以不同的速度运行。这样，可能出现下述三种情况（假定某时刻变量N的值为n）：

1.N=N+1在Print(N)和N=0之前，此时得到的N值分别为n+1，n+1，0

2.N=N+1在Print(N)和N=0之后，此时得到的N值分别为n，0，1.

3.N=N+1在Print(N)和N=0之间，此时得到的N值分别为n，n+1，0

7、试说明PCB的作用具体表现在哪几个方面，为什么说PCB是进程存在的唯一标志？

五个作用：

1、作为独立运行基本单位的标志；

2、实现间断性运行方式；

3、提供进程管理所需要的信息；

4、提供进程调度所需要的信息；

5、实现与其他进程的同步和通信

原因：

当系统创建一个新进程时，就为他创建了一个PCB。进程结束时又回收其PCB，进程也随之消亡。系统是通过PCB感知进程的存在的。

9、进程控制块（PCB）的组织方式有哪几种？

线性方式

解释：将系统中所有的PCB都组织在一张线性表中，将该表的首址存放在内存的一个专用区域中。

优缺点：简单、开销小，但每次查找时都需要扫描整张表

适用于：进程数目不多的系统

链接方式

解释：把具有相同状态进程的PCB分别通过PCB中的链接字链接成一个队列。可以形成就绪队列、若干个阻塞队列和空白队列等。

e.g.对就绪队列而言，往往按进程的优先级将PCB从高到低进行排列，将优先级高的进程PCB排在队列的前面。同样，也可把处于阻塞状态进程的PCB根据其阻塞原因的不同，排成多个阻塞队列，如等待I/O操作完成的队列和等待分配内存的队列等。

索引方式

解释：系统根据所有进程状态的不同，建立几张索引表。如就绪索引表、阻塞索引表等，并把各索引表在内存的首地址记录在内存的一些专用单元中。在每个索引表的表目中，记录具有相应状态的某个PCB在PCB表中的地址

20、试说明线程具有哪些属性？

轻型实体：线程中的实体基本上不拥有系统资源，只有一点必不可少的、能保证独立运行的资源。比如，在每个线程中都应具有一个用于控制线程运行的线程控制块TCB，用于指示被执行指令序列的程序计数器、保留局部变量、少数状态参数和返回地址等的一组寄存器和堆栈。

独立调度和分派的基本单位：在多线程OS中，线程是能独立运行的基本单位，因而也是独立调度和分派的基本单位。由于线程很“轻”，故线程的切换非常迅速且开销小。

可并发执行：一个进程中的多个线程之间可以并发执行，不同进程中的线程也能并发执行。

共享进程资源在同一进程中的各个线程，都可以共享该进程所拥有的资源：这首先表现在所有线程都具有相同的地址空间（进程的地址空间），这意味着，线程可以访问

该地址空间的每一个虚地址；此外，还可以访问进程所拥有的已打开文件、定时器、信号量机构等。

23、何谓用户级线程和内核支持核程？

都是线程的实现方式

	解释	优点	缺点
内核支持性线程 KST	线程的创建、阻塞、撤销和切换都在内核空间实现。为了对内核线程进行控制和管理，在内核空间也为每一个内核线程设置了一个线程控制块，内核根据该控制块而感知某线程的存在并加以控制。 大多数OS都支持 调度以线程为单位进行	1、在多处理器系统中，内核能够同时调度同一进程中多个线程并行执行； 2、如果进程中的一个线程被阻塞了，内核可以调度该进程中的其他线程占用处理器运行，也可以运行其他进程中的线程； 3、内核支持线程具有很小的数据结构和堆栈，线程的切换比较快，切换开销小； 4、内核本身也可以采用多线程技术，可以提高系统的执行速度和效率。	系统开销大：用户进程在用户态运行，而线程调度和管理在内核实现，因此在同一个进程中的线程切换需要从用户态转到内核态进行。
用户级线程 ULT	仅存于用户空间中，与内核无关。 调度以进程为单位进行	1、线程的切换不需要装换到内核空间 2、调度算法可以是进程专用的 3、用户级线程的实现与OS平台无关	1、阻塞问题：当执行一个系统调用时，不仅该线程被阻塞，而且进程内的所有线程都会被阻塞； 2、进程中只有一个线程能执行

26、多线模型有那几种类型？多对一模型有何优缺点？

用户线程模型、内核线程模型和混合线程模型。

多对一模型的优缺点：

优点

1、相对于一对一模型，多对一模型的线程切换要快速许多

2、由于多个用户线程对应一个内核线程，因此用户线程数量几乎没有限制。

缺点

1、如果其中一个用户线程阻塞，那么所有的线程将都无法执行

2、在多处理器系统上，处理器的增多线程性能也不会有明显的帮助

自出题：爸、妈、儿、女共用一个盘子，盘中一次只能放一个水果。当盘子为空时，爸或妈可将一个水果放入果盘中。已知爸只可放苹果，妈只可放橘子。若放入果盘中的是橘子，则允许儿子吃，女儿必须等待；若放入果盘中的是苹果，则允许女儿吃，儿子必须等待。根据题意，用信号量机制写出上述过程的程序。

本题实际上是生产者-消费者问题的一种变形。这里，生产者放入缓冲区的产品有两类，消费者也有两类，每类消费者只消费其中固定的一类产品。

解：在本题中，应设置三个信号量S、So、Sa，信号量S表示盘子是否为空，其初值为l；信号量So表示盘中是否有桔子，其初值为0；信号量Sa表示盘中是否有苹果，其初值为0。同步描述如下：

```
int S＝1;
int Sa＝0;
int So＝0;
main()
{
   Cobegin
   father();   /*父亲进程*/
   son();      /*儿子进程*/
   daughter(); /*女儿进程*/
   coend
}

father()
{
   while (TRUE)
   {
      P(S);
      将水果放入盘中;
      if（放入的是桔子）V(So);
```

```
        else V(Sa);
    }
}
son()
{
    while (TRUE)
    {
        P(So);
        从盘中取出桔子;
        V(S);
        吃桔子;
    }
}
daughter()
{
    while (TRUE)
    {
        P(Sa);
        从盘中取出苹果;
        V(S);
        吃苹果;
    }
}
```

第三章

画重点

调度分为哪几种

名称	调度对象	作用	多道批处理	分时系统	实时系统	备注
高级/长程/作业调度	作业	根据算法，决定将外存上处于后备队列中的哪几个作业调入内存，为它们创建进程、分配必要的资源，并将它们放入就绪队列	√	x	x	周期长，频率低
低级/进程/短程调度	进程/内核级线程	根据算法，决定就绪队列中的哪个进程应获得处理机，并由分派程序将处理机分配给被选中的进程	√	√	√	运行频率最高
中级/内存调度		提高内存利用率和系统吞吐量。把暂时不能运行的进程，调至外存等待，此时为挂起状态。当他们具备运行条件且内存稍有空闲时，把外存上已具备运行条件的就绪进程再重新调入内存，修改其状态为就绪状态。实际上是存储器管理中的对换功能	/	/	/	频率介于两种调度之间

调度算法

- 作业调度
  - 先来先服务 FCFS
  - 短作业优先 SJF
  - 优先级 PSA
  - 高响应比优先 HRRN
- 进程调度
  - 轮转
  - 优先级
    - 抢占式
    - 非抢占式
  - 多队列
  - 多级反馈队列
  - 基于公平原则的
    - 保证
    - 公平分享

先来先服务调度算法 FCFS

经常与其它调度算法相结合使用。

用于作业调度：优先考虑在系统中等待时间最长的作业，不管该作业所需执行时间的长短。将他们调入内存，为他们分配资源和创建进程。然后把它们放入就绪队列。

用于进程调度：从就绪的进程队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或阻塞后，进程调度程序才将处理机分配给其它进程。

短作业优先调度算法 SJF

作业越短，优先级越高（作业长短以作业所要求的运行时间来衡量）。

用于作业调度和进程调度。

四个缺点：

1、必须预知作业的运行时间

2、对长作业非常不利

3、人机无法实现交互

4、不能保证紧迫性作业能得到及时处理

优先级调度算法 PSA

进一步分为{非抢占式 / 抢占式

非抢占式：一旦分配给优先级最高的进程后，该进程一直执行直至完成或因该进程发生某事件而放弃处理机。

抢占式：在执行期间，只要出现了另一个优先级更高的进程，就将处理机分配给他。常用于对实时性要求较高的系统。

优先级的分类{静态优先级 / 动态优先级

静态优先级：创建进程时间时确定。

优点：简单，系统开销小

缺点：不够精确，可能出现优先级低的进程长期不被调度的情况

动态优先级：先赋予优先级，随进程推进或等待时间的增加而改变。

高响应比优先调度算法 HRRN

解释：综合 FCFS 和 SJF，既考虑作业运行时间，又考虑作业等待时间。

实现：

$$优先权 = \frac{等待时间 + 要求服务时间}{要求服务时间} = \frac{响应时间}{要求服务时间} = 响应比R_p$$

可以得出：

等待时间相同时，要求服务时间越短优先权越高。此时类似 SJF。

要求服务时间相同时，优先权决定于等待时间。此时类似 FCFS。

长作业的优先级可以随等待时间的增加而提高。

缺点：每次进行调度之前做响应比的计算，增加系统开销。

多队列调度算法

优点（与其他的不同）：不必事先知道各进程所需的执行时间。公认好。

多级反馈队列调度算法

轮转算法 RR 原理 P93

解释：在分时系统中使用，采取公平的处理机分配方式，让就绪队列上的每个进程每次仅运行一个时间片。

原理：将所有的就绪进程按 FCFS 策略拍成一个就绪队列。系统设置每隔一定时间产生一次中断，去激活进程调度程序进行调度，把 CPU 分配给队首进程，并令其执行一个时间片。运行完毕后把处理机分配给新就绪队列中的队首进程，也让他执行一个时间片。确保就绪队列中的所有进程在确定的时间段内，都能获得一个时间片的处理机时间。

时间片大小：略大于一次典型的交互所需要的时间（过长退化为 FCFS，过短增加系统开销）

死锁

解释：如果一组进程中每一个进程都在等待仅由该组进程中的其它进程才能引发的事件，那么该组进程是死锁的。

两个产生原因

1、竞争资源：当系统中供多个进程共享的资源如打印机、公用队列等，其数目不足以满足诸进程的需要时，会引起诸进程对资源的竞争而产生死锁。

2、进程间推进顺序非法：进程在运行过程中，请求和释放资源的顺序不当。

银行家算法

见作业

作业

30、在银行家算法的例子中，如果p_0发出的请求向量由 Request(0,2,0)改为 Request(0,1,0)，问系统可否将资分配给它？

答：可以

step1.Request0（0,1,0）<=Need(7,4,3);

step2.Request0（0,1,0）<=Available(2,3,0);

step3.系统暂时先假定可为p_0分配资源并修改相关数据

	allocation			Need			Avalilable		
	A	B	C	A	B	C	A	B	C
p_0	0	2	0	7	3	3	2	2	0
p_1	3	0	2	0	2	0			
p_2	3	0	2	6	0	0			
p_3	2	1	1	0	1	1			
p_4	0	0	2	4	3	1			

step4.进行安全检查:

	work			Need			allocation			work+allocation			finish
	A	B	C	A	B	C	A	B	C	A	B	C	
p_1	2	2	0	0	2	0	3	0	2	5	2	2	true
p_3	5	2	2	0	1	1	2	1	1	7	3	3	true
p_0	7	3	3	7	3	3	0	2	0	7	5	3	true
p_2	7	5	3	6	0	0	3	0	2	10	5	5	true
p_4	10	5	5	4	3	1	0	0	2	10	5	7	true

由所进行的安全性检查得知，可以查找一个安全序列$\{p_1, p_3, p_0, p_2, p_4\}$。因此，系统是安全的，可以立即将$p_0$所申请的资源分配给它。

30、在银行家算法的中，若出现下述资源分配情况，试问：

（1）该状态是否安全？

（2）若进程p_2提出请求 Request(1,2,2,2)后，系统能否将资源分配给它？

process	allocation	need	available
p_0	0032	0012	1622
p_1	1000	1750	
p_2	1354	2356	
p_3	0332	0652	
p_4	0014	0656	

(1)

安全，安全序列$\{p_0, p_3, p_4, p_1, p_2\}$。

(2)

step1.Request(1,2,2,2)<=Need(2,3,5,6);

step2.Request(1,2,2,2)<=Available(1,6,2,2);

step3.系统暂时先假定可为p_2分配资源并修改相关数据

process	allocation	need	available
p_0	不变	不变	不变
p_1	不变	不变	不变
p_2	2576	1134	0400
p_3	不变	不变	不变

step4.进行安全检查:available(0,4,0,0)已经不能再满足任何进程的需要，所以系统进入不安全状态，系统不将资源分配给他

第四、五章

画重点

存储器层次

至少三层，还可细分为6层

- CPU寄存器{寄存器
- 主存
  - 高速缓存
  - 主存储器
  - 磁盘缓存
- 辅存
  - 固定磁盘
  - 可移动存储介质

层次↑，越靠近 CPU，价格↑，速度↑，相对所配置的存储容量↓

分区分配算法

（箭头的意思是随各种情况的发展）

- 单一连续分配
- ↓ *为适应OS由单道向多道发展*
- 固定分区分配
- ↓ *为更好地适应不同大小的用户程序要求*
- 动态分区分配
  - 基于顺序搜索
    - 首次适应算法FF
    - 循环首次适应算法NF
    - 最佳适应算法BF
    - 最坏适应算法WF
  - 基于索引搜索
    - 快速适应算法
    - 伙伴系统
    - 哈希算法
- 动态可重定位分区分配
- ↓ *为更好地提高内存的利用率*
- 离散分配
  - 分页存储管理方式
  - 分段存储管理方式
  - 段页式存储管理方式

单一连续分配

单道程序环境使用。

内存分为系统区&用户区，系统区仅供OS使用，用户区仅装有一道用户程序，独占整个内存的用户空间。

固定分区分配

多道程序使用。

将整个用户空间划分为若干个固定大小的区域，每个分区只装入一道作业。

现在已经很少用了，在某些用于控制多个相同对象的控制系统中会用

动态分区分配/可变区分配

根据进程的实际需要，动态的为之分配内存空间。实现时会涉及三方面的问题：

1、分配中所用的数据结构

2、分区分配算法

3、分区的分配与回收操作

下面展开说说1和2。

1、分配中所用的数据结构{空闲分区表 / 空闲分区链

分区号	分区大小(KB)	分区始址(K)	状态
1	50	85	空闲
2	32	155	空闲
3	70	275	空闲
4	60	532	空闲
5	…	…	…

图 4-6　空闲分区表

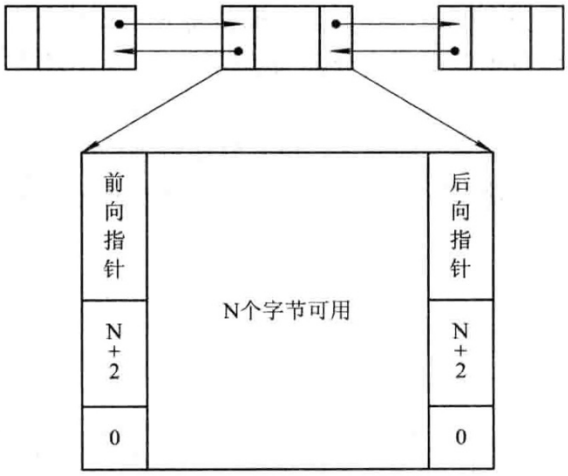


图 4-7　空闲链结构

2、分区分配算法

	算法名	解释	优点	缺点
基于顺序搜索（适用	首次适应算法 FF	空闲分区链以地址递增的次序链接。 step1.从链首开始查找，找到一个大小能满足要求的空闲分区为止。 step2.按照作业大小，从该分区中划出一块内存空间，分配给请求者，余下的空闲分区仍留在空闲链内	倾向于优先利用内存中低址部分的空闲分区，从而保留高址部分的大空闲区，这为以后到达的大作业分配大的内存空间创造了条件	低址部分不断被划分，会留下许多碎片。每次查找又是从低址开始，会增加开销。

于不太大的系统）	循环首次适应算法NF	step1从上次找到的空闲分区的下一个空闲分区开始查找 step2.从中划出一块与请求大小相等的内存空间分配给作业	使内存中的空闲分区分布的更均匀，从而减少查找空闲分区时的开销	缺乏大的空闲分区
	最佳适应算法BF	所有空闲分区按其容量以从小到大的顺序形成一空闲分区链，第一次找到的能满足要求的空闲区一定是最佳的		每次分配后所切割下来的剩余部分总是最小的，这样，在存储器中会留下许多难以利用的碎片。
	最坏适应算法WF	扫描整个空闲分区表或链表时，总是挑一个最大的空闲区	1、产生碎片的可能性最小 2、对中小作业有利 3、查找效率高	
基于索引搜索（适用于大中型系统）	快速适应算法	将空闲分区根据其容量大小进行分类，对于每一类具有相同容量的所有空闲分区单独设立一个空闲分区链表。 同时在内存中设立一张管理索引表，其中的每一个索引表项对应了一种空闲分区类型，并记录该类型空闲分区链表表头的指针	1、查找效率高 2、不会产生内存碎片	1、为了有效合并分区，在分区归还主存时的算法复杂，系统开销大 2、分配空闲分区时以进程为单位，或多或少存在浪费，以空间换时间
	伙伴系统			

	哈希算法	构造一张以空闲分区大小为关键字的哈希表，该表的每一个表项纪录了对应的空闲分区链表表头指针。 分配时，根据所需空闲分区大小，通过哈希函数计算，得到在哈希表中的位置，得到相应的空闲分区链表		

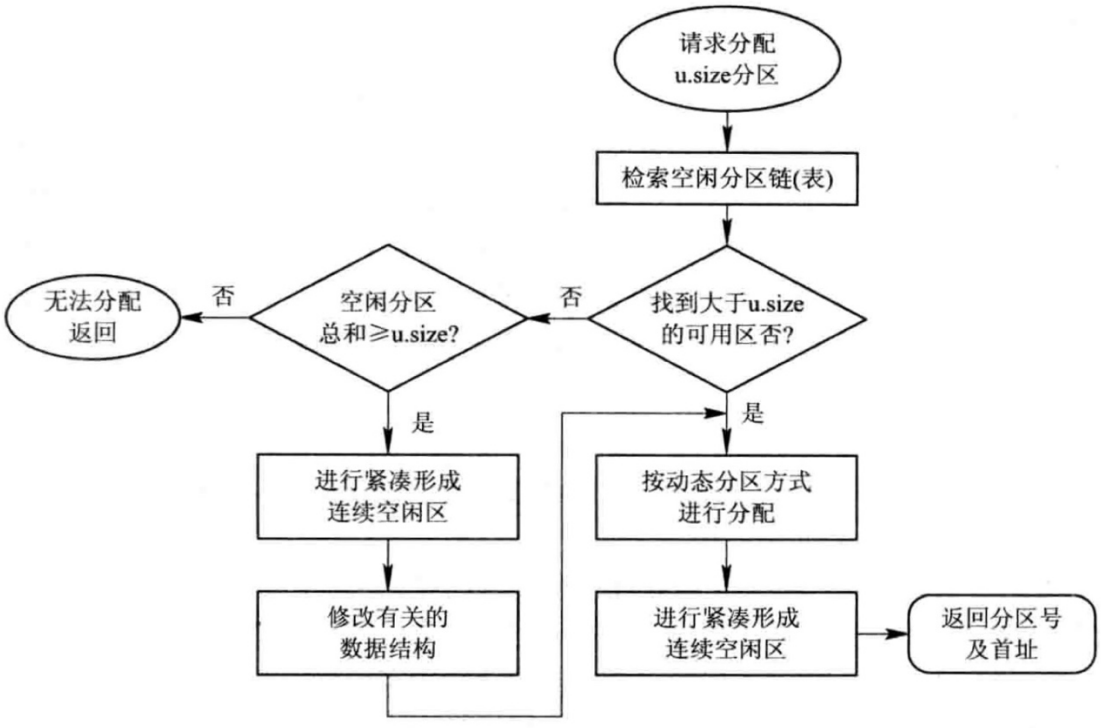
动态可重定位分区分配

在动态分区分配算法上增加了紧凑的功能

紧凑的is：通过移动内存中作业的位置，把原来多个分散的小分区拼接成一个大分区的方法。

在系统中增设一个重定位寄存器（硬件），用以存放程序在内存中的起始地址。程序执行时真正访问的内存地址是相对地址与重定位寄存器中的地址相加而成。

地址变换过程是在程序执行期间，随着对每条指令或数据的访问自动进行的。



碎片的产生

解释：难以利用，很小的空闲分区

见BF的缺点

分页存储原理，分段存储原理

根据离散分配方式所分配地址空间的基本单位的不同，进一步分为

- 分页存储管理方式
- 分段存储管理方式
- 段页式存储管理方式

下面展开说说分页存储原理和分段存储原理。

分页存储原理

页：将用户程序的地址空间分为若干个固定大小的区域，称为**页**/页面，典型页面大小为应是2的幂，通常为1KB～8KB。

块：将内存空间分为若干个物理**块**或页框。

页和块大小相同。

这样可将用户程序的任一页放入任一物理块中。

分页地址中的**地址结构**：

31 ... 12	11 ... 0
页号P	位移量W(页内地址)

如图所示的意思：每页大小为4KB（2^{12}），地址空间最多允许有1M（2^{20}）页

设逻辑地址空间中的地址为A，页面大小为L，页号为P，页内地址为d，则有以下关系：

$$P=int（A/L）$$

$$d=A \bmod L$$

例如，L=1KB，A=2170B，那么P=int(2170/1024)=2，d=2170 mod 1024=122

页表：实现从页号到物理块号到地址映射

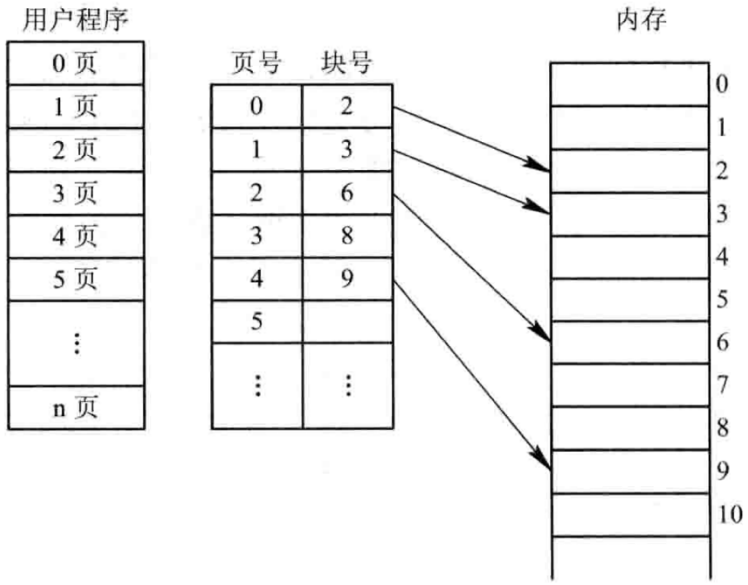


图 4-14　页表的作用

分段存储原理

所有存储管理方式的基础

五个优点：

1、方便编程

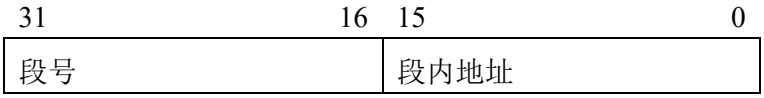
2、信息共享

3、信息保护

4、动态增长

5、动态链接

分段地址中的**地址结构：**



如图所示的意思：允许一个作业最长有64K个段（对应图中段号），每个段的最大长度为64KB（对应图中段内地址）

段表：实现从逻辑段道物理内存区的映射

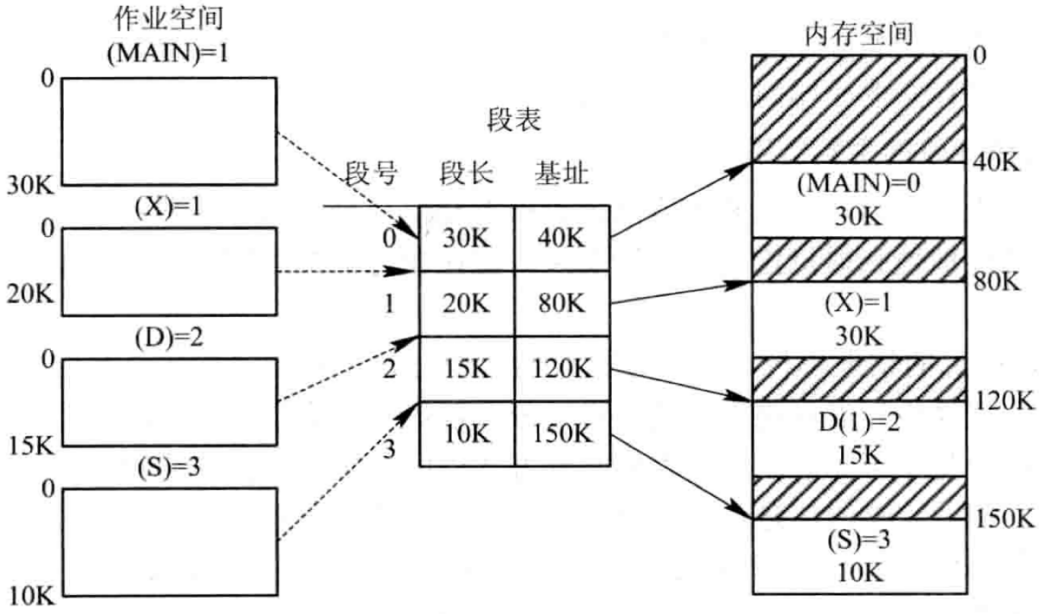


图 4-19 利用段表实现地址映射

虚拟存储概念及特征

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一种存储器系统

三个特征：

1、多次性：相对于传统存储器管理方式的一次性而言，一个作业中的程序和数据无需在作业运行时一次性全部装入内存，而是可以被多次调入内存运行，即只需将当前要运行的那部分程序和数据装入内存即可开始运行。

2、对换性：相对于传统存储器管理方式的常驻性而言，指一个作业中的程序和数据，无须在作业运行时一直常驻内存，而是允许在作业的运行过程中进行换进换出。

3、虚拟性：能够从逻辑上扩充内存容量，使用户看到的内存容量远大于实际内存。

作业

某系统采用分页存储管理方式，拥有逻辑空间 32 页，每页 2K，拥有物理空间 1M，则逻辑地址共（16）位，其中页号占（5）位，页内地址占（11）位；若不考虑访问权限等，进程的页表项最多有（32）项，每项至少（9）位。

（多选题）上题中如果物理空间减少一半，则页表项（B），每项长度（D）。

A.减少一半

B.不变

C.可减少一半

D.可减少一位

（多选题）与传统的存储器管理方式比较，虚拟存储器具有以下重要特征（BCD）

A.驻留性

B.多次性

C.对换性

D.虚拟性

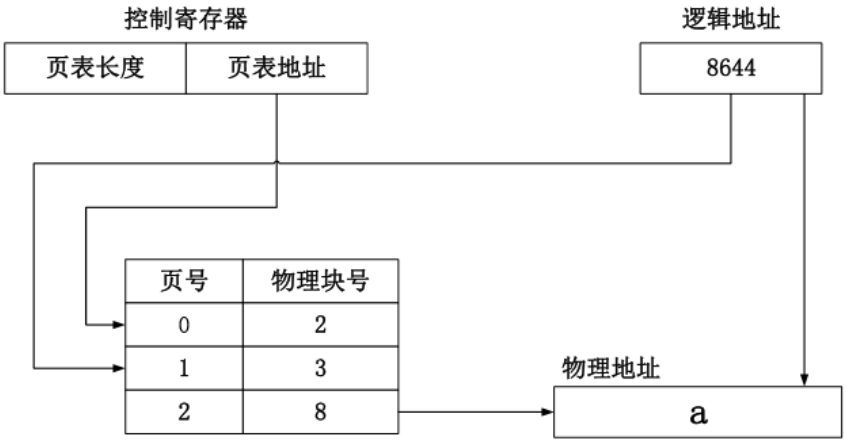
页式存储系统的逻辑地址是由页号和页内地址两部分组成。假定页面的大小为 4K，地址变换过程如图所示，图中逻辑地址用十进制表示，图中有效地址经过变换后，十进制物理地址 a 应为（ ）

A.33220

B.33221

C.45481

D.25000



某存储器的用户编程空间共 32 个页面，每页为 1KB，内存为 16KB。假定某时刻一用户页表中已调入内存的页面的页号和物理块号的对照表如下

页号	物理块号
0	7
1	13
2	5
3	9

则逻辑地址 0D35(H)所对应的物理地址是什么？(要求写出计算步骤)