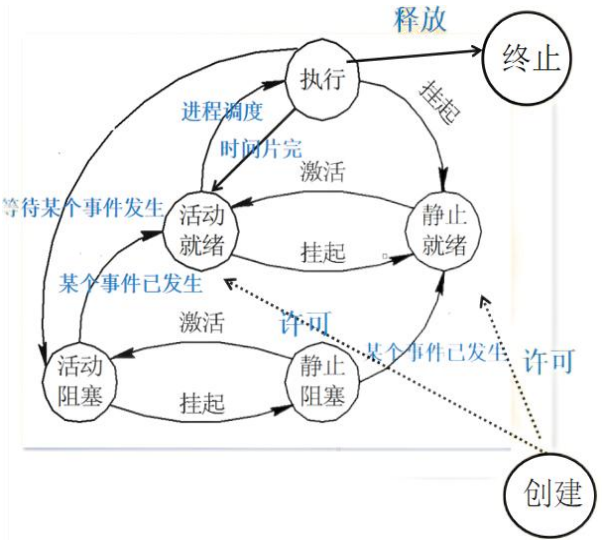
《计算机操作系统》知识点

目录

1、操作系统的目标：方便性、有效性（由资源利用率、系统吞吐量恒定）、可扩充性和开放性。
2、操作系统的作用：用户与计算机硬件系统之间的接口（命令方式、系统调用方式、图形化窗口方式）、系统资源的管理者、对计算机资源的抽象。
3、推动操作系统发展的主要动力：不断提高计算机资源利用率、方便用户、器件的不断更新迭代、计算机体系结构的不断发展、不断提出新的应用需求。
4、操作系统的发展过程：人工操作方式、脱机输入/输出方式、单道批处理系统、多道批处理系统、分时系统、实时系统、微机操作系统、网络操作系统等。
5、操作系统的基本特征：并发、共享、虚拟、异步
6、并发、并行性：并发性是指两个或多个事件在同一时间间隔内发生；并行性是指两个或多个事件在同一时刻发生。
7、什么是微内核 OS 结构：尽最大努力剔除内核的多余成分，并将他们移到核外子系统中进行实现。内核只实现一些必要的、最基本的功能，保持内核的简洁高效。
8、微内核 OS 结构的优点：
- 提高了系统的可扩展性。
- 增强了系统的可靠性。
- 可移植性强。
- 提供了对分布式系统的支持。
- 融入了面向对象技术。

缺点：运行效率有所降低。
9、并发执行的特点：间断性、失去封闭性、不可再现性。
10、为什么会出现进程、什么是进程、进程的特点？
目的：为了能使程序并发执行，并且可以对并发执行的程序加以描述和控制，人们引入了进程的概念。
定义：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。
特点：动态性、并发性、独立性、异步性。
11、进程挂起和激活
引入挂起状态后，进程状态的转换
（1） 活动就绪→静止就绪
（2） 活动阻塞→静止阻塞
（3） 静止就绪→活动就绪
（4） 静止阻塞→活动阻塞
（5） 创建→活动就绪
（6） 创建→静止就绪



12、PCB 进程控制块的内容？
PCB 的作用：
（1）作为独立运行基本单位的标志。
（2）能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与通信。
PCB 中的信息：
（1）进程标识符
（2）处理机状态

（3）进程调度信息
（4）进程控制信息
PCB 的组织方式：
（1）线性方式
（2）链接方式
（3）索引方式
13、进程控制：进程控制一般是由 OS 内核中的原语来实现的。
14、临界区（临界资源）：硬件资源：一段时间内只允许一个进程使用的资源。软件资源：多个进程共享的变量。访问临界资源的那段代码称为临界区。
15、进程同步的两种形式的制约关系：
（1）间接相互制约关系：由于需要共享系统资源，必须保证多个进程对之只能互斥地访问。
（2）直接相互制约关系：进程将为完成同一项任务而相互合作。
16、同步机制应遵循的规则（16 字准则）：
（1）空闲让进：临界区无进程，应允许一个请求进入临界区的进程立即进去自己的临界区
（2）忙则等待：临界区有进程，让其它试图进入临界区的进程必须等待
（3）有限等待：保证在有限的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：当进程不能进入自己的临界区时，应立即释放处理机
17、低级通信和高级通信（信号量机制属于低级通信）
按数据量的大小划分：低级通信（传送数据量少）、高级通信（传送大量数据）
高级通信的类型：
（1）共享存储器系统
（2）管道通信系统（半双工方式-单向、以字符流形式传送数据）
（3）消息传递系统（直接消息传递：信息缓冲机制、间接消息传递：信箱通信方式）
（4）客户机-服务器系统
18、什么是管道：是指用于连接一个读进程和一个写进程以实现它们之间通信的一个共享文件，又名 pipe 文件。管道机制必须提供以下三方面的协调能力：互斥、同步、确定对方是否存在。
19、原语：原语是系统状态下执行的某些具有特定功能的程序段，一旦开始执行就不能被中断，否则会产生系统混乱。
20、信箱的类型及创建者：
（1）私用邮箱，由用户创建
（2）公用邮箱，由操作系统创建
（3）共享邮箱，由某进程创建
21、为什么引入线程？
提高并发执行的程度，进一步提高资源的利用率和系统吞吐量。
22、线程的类型：
（1）内核支持线程 KST（由系统控制）
优点：①多处理机系统中，内核能同时调度同一个进程中的多个线程并发执行；②如果一个进程中的一个线程被阻塞，内核可以调度该进程中其他线程运行，或者运行其他进程中的线程；③内核支持线程具有较小的数据结构和堆栈，线程切换开销小；④内核本身也可以采用多线程技术，可提高系统的执行速度和效率。
缺点：同一用户进程中，从一个线程切换到另一个线程时需要从用户态切换到内核态，开销较大。
（2）用户级线程 ULT（由用户控制）

优点：①线程切换不需要转换到内核空间；②调度算法可以是进程专用的；③用户线程的实现与 OS 平台无关。

缺点：①系统调用的阻塞问题。当线程执行一个系统调用时，该进程以及进程内所有线程均被阻塞；②多线程不能利用多处理机进行多重处理。

（3）组合方式（形成了三种不同的模型）

◆多对一模型：多个用户级线程映射到一个内核级线程，线程管理在用户空间完成。此模式中，用户级线程对操作系统不可见。

优点：线程管理在用户空间进行，效率高。

缺点：一个线程使用内核服务时被阻塞，整个进程都被阻塞。多个线程不能并行运行在多处理机上。

◆一对一模型：每个用户级线程映射到一个内核级线程。

优点：当一个线程被阻塞时，允许另一个线程继续运行，并发能力强。

缺点：每创建一个用户级线程都需要创建一个内核级线程与其对应，创建线程的开销大，会影响到应用程序的性能。

◆多对多模型：将 n 个用户级线程映射到 m 个内核级线程（m<=n）

特点：克服了多对一并发度不高的缺点，又克服了一对一模型中一个用户进程占太多内核级线程而开销大的缺点。

23、处理机的调度层次：

（1）高级调度（长程调度、作业调度）：调度对象是作业，根据某种算法，将外存上处于后备队列中的哪几个作业调入内存，创建进程、分配资源、放入就绪队列。主要用于：多道批处理系统。

（2）低级调度（进程调度、短程调度）：调度对象是进程（或内核级线程），根据某种算法，觉得就绪队列中的那个进程获得对应的处理机、并将处理机分配给选中的进程。主要用于：多道批处理系统、分时系统、实时系统。

（3）中级调度（内存调度、交换调度、中程调度）：提高内存利用率和系统吞吐量，调度对象也是进程，从外存到内存（已经就绪具备运行条件），从内存到外存（暂时不能运行的）。

特点：

低级调度（进程调度）：频率最高。

高级调度（作业调度）：周期较长、运行频率低。

中级调度：频率介于以上两者之间。

24、进程的调度方式：

（1）抢占式：允许调度程序根据某种原则，去暂停某个正在执行的进程，将已分配给该进程的处理机重新分配给另一进程。遵循以下原则：优先权原则、短进程优先原则、时间片原则。

（2）非抢占式：一旦把处理机分配给某进程后，就让它一直运行下去，绝不会因为时钟中断或任何其它原因去抢占当前正在运行进程的处理机，直至该进程完成，或阻塞时才把处理机分配给其他进程。

25、什么是死锁？

多个进程因竞争资源而造成的一种僵局，若无外力作用，这些进程将无法再向前推进。如果一组进程中的每一个进程都在等待仅由该组进程中的其他进程才能引发的事件，那么该组进程是死锁的。

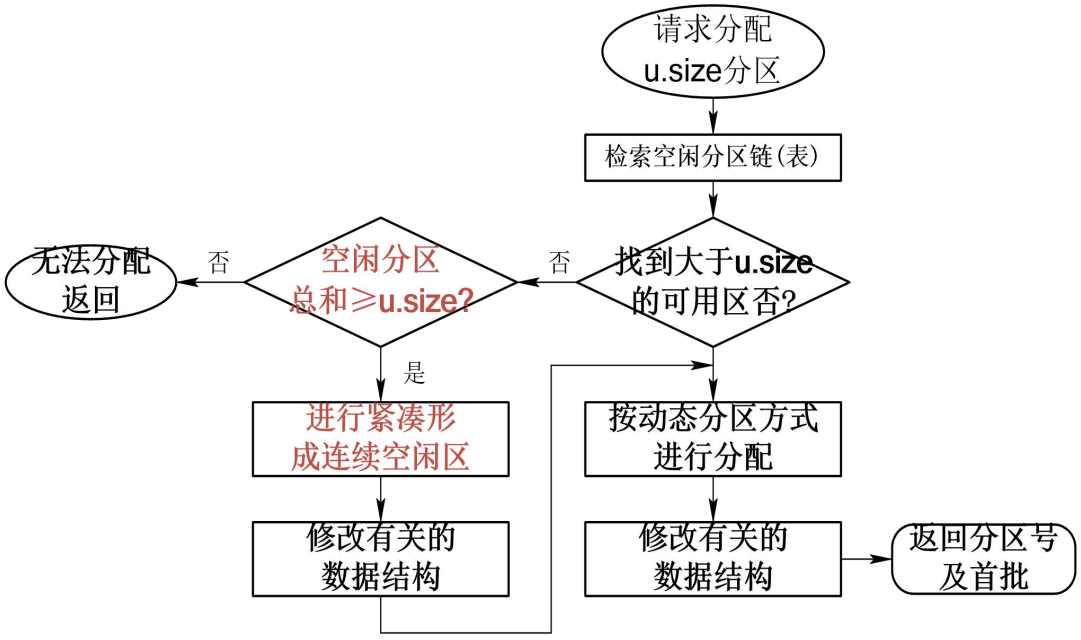
26、产生死锁的原因、产生条件、处理死锁、预防死锁、避免死锁。

◆　产生死锁的原因：

（1）资源竞争引起死锁（竞争不可抢占性资源、竞争可消耗性资源）

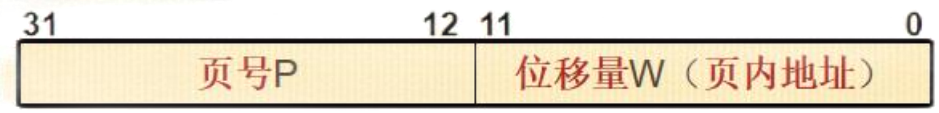
（2）进程推进顺序不当引起死锁
（3）信号量使用不当引起死锁
◆ 产生死锁的必要条件：
（1）互斥条件
（2）请求和保持条件
（3）不可抢占条件（不可剥夺条件）
（4）循环等待条件（环路等待条件）
◆ 处理死锁的方法
（1） 预防死锁
（2） 避免死锁
（3） 检测死锁
（4） 解除死锁
（5） 鸵鸟算法（不理睬死锁）
◆ 预防死锁的方法
（1）破坏“请求和保持”条件：一次性的申请全部资源；允许一个进程只获得初期所需的资源，运行过程再逐步释放已分配给自己的、且用完的资源，再请求新的所需资源。
（2）破坏“不可抢占”条件
（3）破坏“循环等待”条件：有序分配法（规定每个进程必须按序号递增的顺序请求资源）
◆ 避免死锁的方法
（1）使系统保持在安全状态，运行进程是安全序列
（2）使用银行家算法避免死锁
27、怎么去检测死锁、解除死锁？
检测死锁：
（1）资源分配图
（2）死锁定理
解除死锁：
（1）立即结束所有进程的执行，并重新启动操作系统。
（2）资源剥夺法：当发现死锁后，从其它进程剥夺足够数量的资源给死锁进程，以解除死锁状态。
（3）撤消进程法：采用强制手段从系统中撤消一个/一部分死锁进程，并剥夺这些进程的资源供其它进程使用。（撤销进程的数目最小、撤销进程所付出的代价最小）
28、程序的装入方式，什么时候进行地址转移？
（1）绝对装入方式（只适用于单道程序环境，编译时进行地址转换）
（2）静态重定位方式（程序装入时进行地址转换）缺点：不能在移动，主存利用率低。
（3）动态重定位方式（程序运行时进行地址转换）特点：需要重定位寄存器的支持
29、内存的分配
（1）单一连续分配
将内存分为系统区（内存低端，分配给 OS 用）和用户区（内存高端，分配给用户用）。采用静态分配方式，即作业一旦进入内存，就要等待它运行结束后才能释放内存。
主要特点：管理简单，但因内存中只装入一道作业运行，内存空间浪费大，各类资源的利用率也不高。最简单的一种存储管理方式，仅适用与单用户、单任务的操作系统。
（2）固定分区分配
将内存划分为若干个固定大小（大小相等或者大小不等）的区域，分区一旦划定，在整个执行过程中就不能改变。每个作业占用一个分区，每个分区都是一片连续的内存区域。固定分

区分配是最早使用的一种可运行多道程序的存储管理方法。
主要特点：管理简单，但因作业的大小并不一定与某个分区大小相等，从而使一部分存储空间被浪费。所以主存的利用率不高。
（3）动态分区分配
动态分区分配是一种动态划分存储器的分区方法。
存储管理方法：根据进程的实际需要，动态地为之分配内存空间，并使分区的大小正好适应进程的需要。
主要特点：管理简单，进程的大小与某个分区大小相等，从而主存的利用率有所提高。
常用的数据结构有以下两种形式：空闲分区表、空闲分区链。
（4）动态重定位分区分配
因为动态分区分配会产生一些外部碎片，动态重定位就采用紧凑拼接技术，将内存中所有作业移到内存一端，（即动态重定位），使本来分散的多个小空闲分区连成一个大的空闲区。
主要特点：
可充分利用存储区中的“零头/碎片”，提高主存的利用率。 但若 “零头/碎片”太多，则拼接频率过高会使系统开销加大。
30、动态分区分配算法
（1）首次适应算法（first fit，FF）
实现思路：要求空闲分区表/链以地址递增的次序链接空闲分区。从表/链首开始查找，直至第一个满足要求的空闲分区。按作业的大小，从该分区中划出一块空间，余下部分留在空闲表/链中。
特点：优先利用内存低地址部分的空闲分区，从而保留了高地址部分的大空闲区。但由于低地址部分不断被划分，致使低地址端留下许多难以利用的很小的空闲分区(碎片或零头)，而每次查找又都是从低地址部分开始，增加了查找可用空闲分区的开销。
（2）循环首次适应算法（next fit，NF）
实现思路：要求空闲分区表/链以地址递增的次序链接空闲分区，从上次找到空闲分区的下一个空闲分区开始查找，直至找到一个能满足要求的空闲分区。按作业的大小，从该分区中划出一块空间，余下部分留在空闲表/链中。
特点：使存储空间的利用更加均衡，不致使小的空闲区集中在存储区的一端，但这会导致缺乏大的空闲分区。
（3）最佳适应算法（best fit，BF）
实现思路：要求空闲分区表/链以空闲分区容量以小到大的次序链接空闲分区，从表/链首开始查找，直至找到一个能满足要求的空闲分区。按作业的大小，从该分区中划出一块空间，余下部分留在空闲表/链中。
特点：空闲区一般不可能正好和它申请的内存空间大小一样，因而将其分割成两部分时，往往使剩下的空闲区非常小，从而在存储器中留下许多难以利用的小空闲区（外部碎片）。
（4）最坏适应算法（worst fit，WF）
实现思路：要求空闲分区表/链以空闲分区容量以大到小的次序链接空闲分区，从表/链首开始查找，直至找到一个能满足要求的空闲分区。按作业的大小，从该分区中划出一块空间，余下部分留在空闲表/链中。
特点：总是挑选满足作业要求的最大的分区分配给作业。这样使分给作业后剩下的空闲分区也较大，可装下其它作业，对小作业有利。但由于最大的空闲分区总是因首先分配而划分，当有大作业到来时，其存储空间的申请往往会得不到满足。
31、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，差别在于：增加了紧凑的功能。



32、分页存储管理

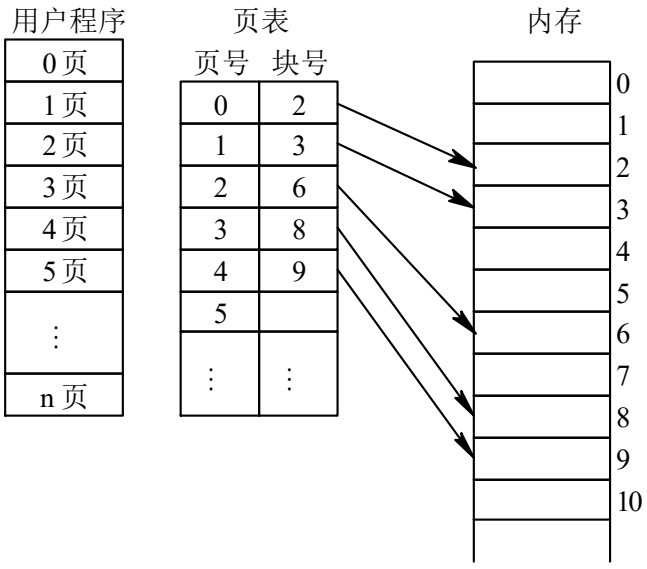
地址结构：



包含两部分内容：页号 P 和偏移量 W（页内地址），图中的地址长度为 32 位，其中 0～11 位为偏移量（页内地址），即每页的大小为：2^{12}B（4KB）；12～31 位为页号，地址空间最多允许2^{20}B（1M）页。

页表：

为了便于在内存找到进程的每个页面所对应的物理块，系统为每个进程建立一张页面映象，简称页表，如图。



（1）作用是实现从页号到物理块号的地址映射。

（2）页表一般存放在内存中。每个进程至少拥有一个页表，其大小由进程长度决定

（3）进程被调度到时，其页表的基址及长度装入页表寄存器。

（4）访问一个字节的数据/指令需访问内存 2 次，所以出现内存访问速度降低的问题。

求页号和偏移量：P = A/L，d = A%L，其中 A 是逻辑地址，L 是页面大小（注意单位统一）。

求某程序的物理地址：

物理地址=物理块号*物理块大小（页大小）+页内地址

（在页表中根据页号找到块号，在使用公式算出物理地址）

33、内存的访问时间（EAT）

访问数据要访问内存两次：①访问页表，求绝对地址（物理地址）。②根据绝对地址访问内存中的数据运行。

无快表时的访问时间：

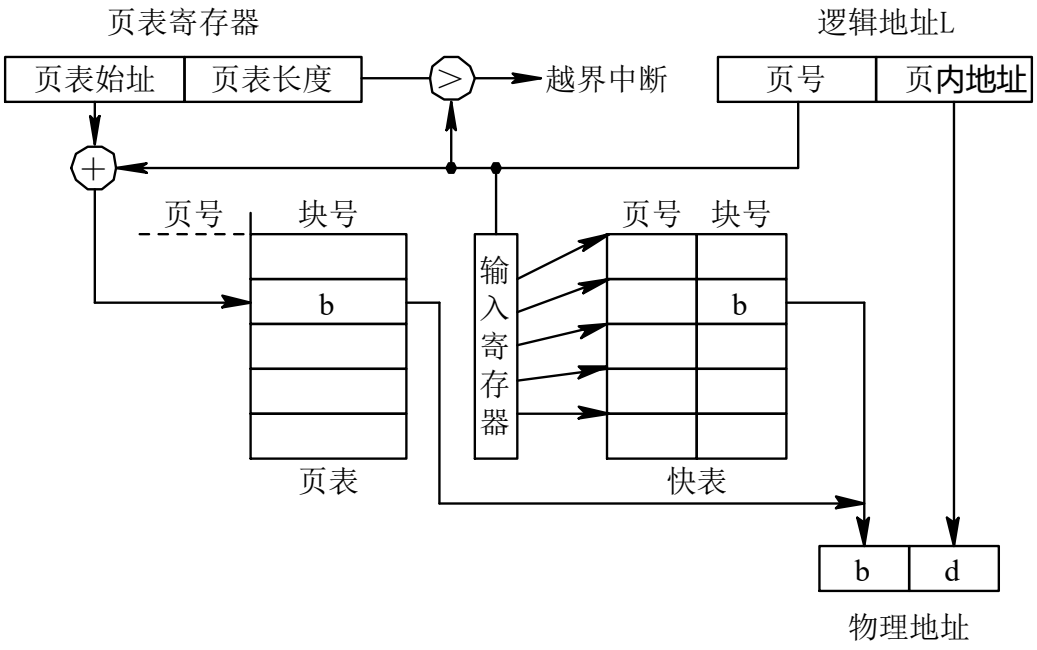
$$EAT = t + t = 2t$$

有快表时的访问时间：

$$EAT = a \times (\lambda + t) + (1 - a)(\lambda + 2t)$$

注：t：一次查页表的时间，λ：一次查快表的时间，a：命中率

引入快表的地址变化机构：

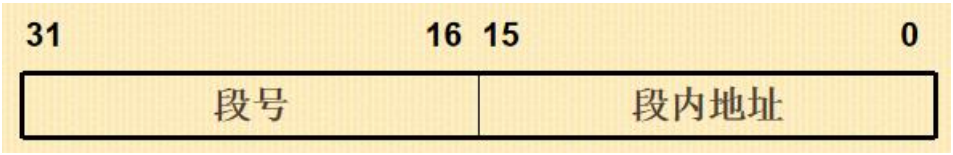


34、为什么会有多级页表？

若逻辑地址空间很大，则划分的页就很多，页表就很大，其占用的存储空间（要求连续）就大，实现较难。解决页表过长的问题，避免了使用大片的连续存储空间。

35、分段存储管理

地址结构：



包含两部分内容：段号和偏移量（段内地址），图中的地址长度为 32 位，其中 0～11 位为偏移量（段内地址），即每段的大小为：2^{16}B（64KB）；16～31 位为页号，地址空间最多允许2^{16}B（64K）个段。

段表：

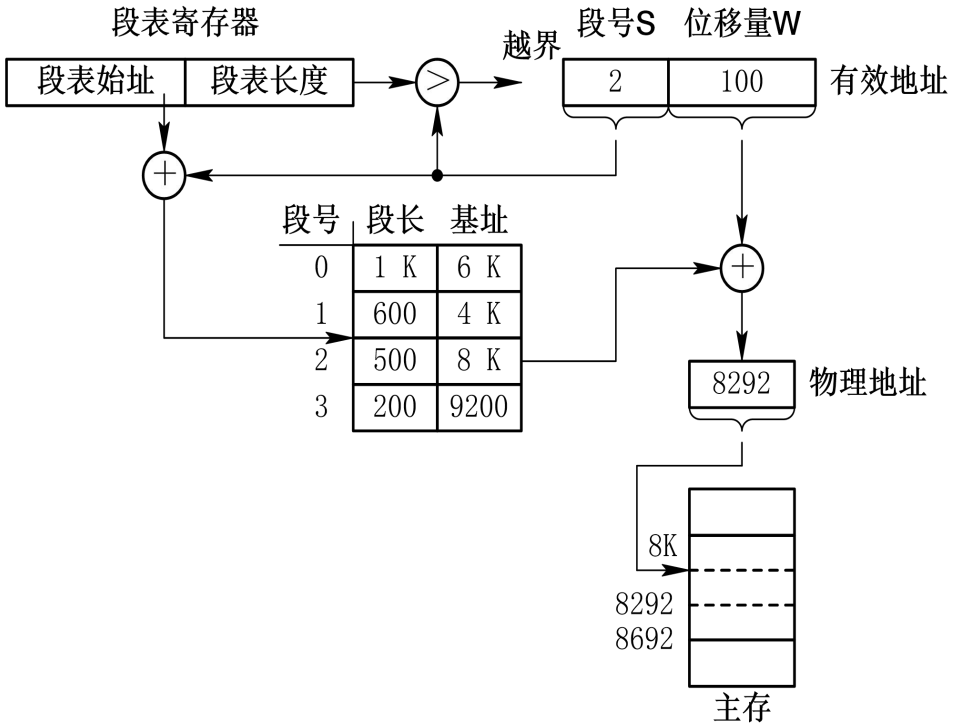
（1）记录了段与内存位置的对应关系

（2）段表常保存在内存中

（3）段表在内存中的基址及长度由段表寄存器给出

（4）访问一个字节的数据/指令需访问内存 2 次，所以也出现内存访问速度降低的问题。为了提高对段表的存取速度，增设一个联想寄存器，利用高速缓冲寄存器保存最近常用的段表项。

地址变换机构：将逻辑地址变换为物理地址的功能。



36、分段分页的主要区别：

相似处：离散分配方式，通过地址映射机构实现地址变换。

区别：

	页式存储管理	段式存储管理
目的	更好满足系统需要	更好满足用户需要
单位	页（物理单位）	段（逻辑单位）
大小	固定（由系统定）	不定（由用户程序定）
内存分配单位	页	段
作业地址空间	一维	二维
优点	有效解决了碎片问题，有效提高内存的利用率	更好地实现数据共享与保护，段长可动态增长，便于动态链接

37、虚拟存储器

虚拟存储器指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，其逻辑容量由内存容量和外存容量之和决定，其运行速度接近于内存，而每位的成本接近于外存。

特征：多次性、对换性、虚拟性。

38、请求分页系统

在分页系统的基础上，增加了请求调页功能、页面置换功能所形成的页式虚拟存储器系统。

硬件支持：

（1）请求页表机制

字段：（新增字段已标红）

页号	物理块号	状态位 P	访问字段 A	修改位 M	外存地址

①状态位 P：用于指示该页是否已调入内存，供程序访问时参考。

②访问字段 A：提供给置换算法在选择换出页面时的参考（用于淘汰访问次数少的）。

③修改位 M：标识该页调入内存后是否被修改过。

④外存地址：指出该页在外存上的地址，通常是物理块号。

（2）缺页中断机制

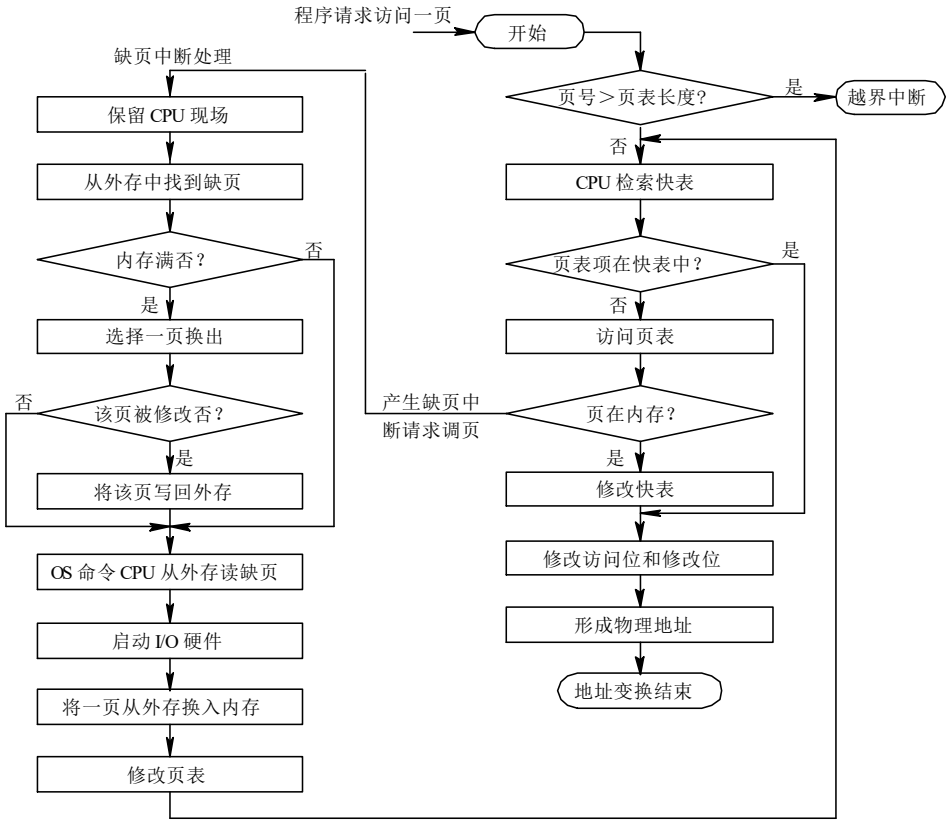
作用：在请求分页系统中，当访问的页不在内存，便产生一缺页中断，请求 OS 将所缺页调入内存空闲块，若无空闲块，则需置换某一页，同时修改相应页表表目。

缺页中断与一般中断的区别：

- 指令执行期间产生和处理中断信号；
- 一条指令在执行期间，可能产生多次缺页中断。

（3）地址变换机制

请求分页中的地址变换机构：



39、页面调入策略

（1）何时调入页面：

采取预调页策略或请求调页策略。

①预调页策略：将那些预计在不久便被访问的页预先调入内存。这种调入策略提高了调页的效率，减少了 I/O 次数。但由于这是一种基于局部性原理的预测，若预调入的页面在以后很少被访问，则造成浪费，故这种方式常用于程序的首次调入。

②请求调页策略：当进程运行中访问的页出现缺页时，则发出缺页中断，提出请求调页，由 OS 将所需页调入内存。这种策略实现简单，应用于目前的虚拟存储器中，但易产生较多的缺页中断，且每次调一页，系统开销较大，容易产生抖动现象。

（2）何处调入页面：

①系统拥有足够的对换区空间，运行前可将与进程相关的文件从文件区复制至对换区，从对换区调入页面。以后缺页时，全部从对换区调页。

②系统缺少足够的对换区空间，凡是不会被修改的文件，直接从文件区调入页面。对可能会修改的文件换出时换至对换区，以后从对换区调页。

③UNIX 方式，凡未运行过的页面均从文件区调页，运行过的页面和换出的页面均从对换区调页。

40、抖动

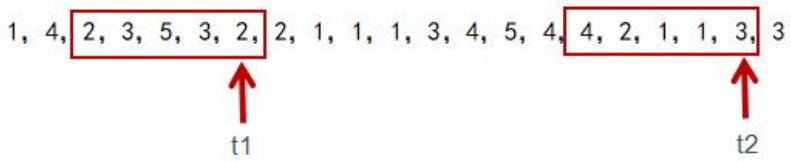
什么是抖动：每个进程的大部分时间都用于页面的换进和换出，而不能再去做其他有效的工作，从而导致发生处理机的利用率急剧下降并趋于 0 的情况。

发生抖动的原因：在系统中运行的进程太多，分配给每个进程的物理块太少，不能满足进程正常运行的基本要求，致使每个进程在运行时，频繁的出现缺页。

41、工作集：

在某段时间间隔内，进程要访问的页面集合（在时间 t 之前的一段时间△内该进程所引用的全部页面的集合）

例：某进程对页面的访问次序如下，系统为该进程设定的工作集窗口为 5，在 t1 时刻，进程的工作集为{2，3，5}，t2 时刻，进程的工作集为{1，2，3，4}



为了防止抖动现象，分配给进程的物理块数要大于工作集的大小。

42、抖动的预防

（1）采取局部置换策略

（2）把工作集算法融入到处理机调度中

（3）利用“L（缺页的平均时间）=S（置换一个页面的时间）”准则调节缺页率

（4）选择暂停的进程

43、I/O 层次

（1）用户层 I/O 软件：产生 I/O 请求、格式化 I/O、Spooling

（2）设备独立性软件：映射、保护、分块、缓冲、分配

（3）设备驱动程序：设置设备寄存器、检查设备寄存器状态

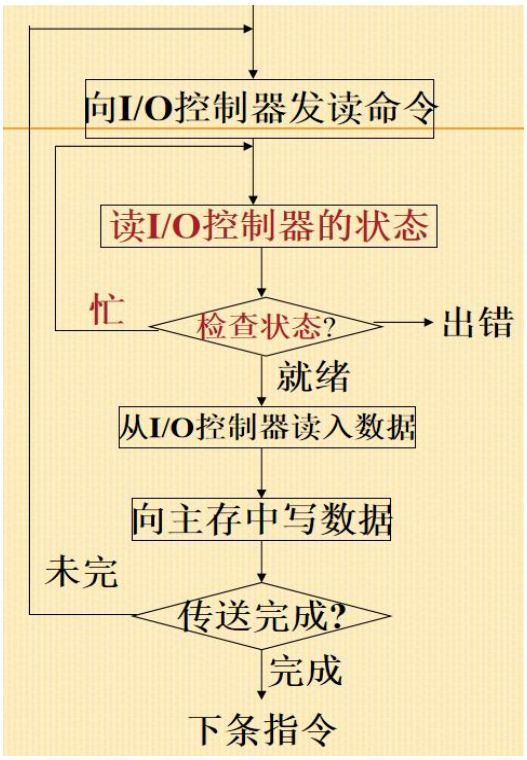
（4）中断处理程序

（5）硬件：执行 I/O 操作

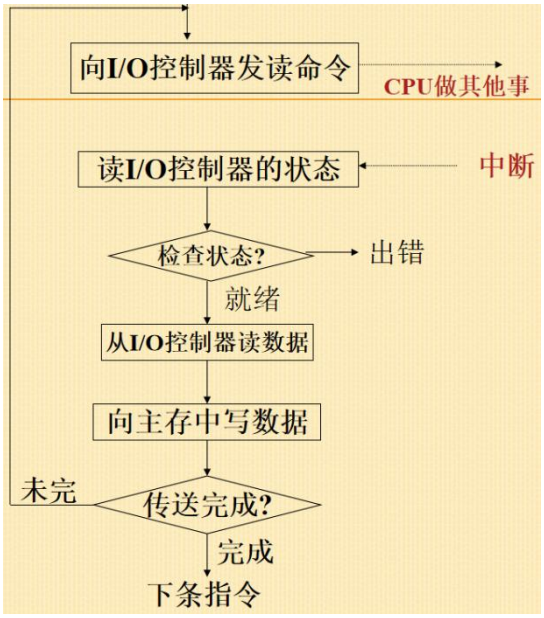
44、设备控制器的作用（基本功能）

（1）接受和识别命令
通过控制器中的控制寄存器来存放和接受 CPU 发来的命令和参数。
（2）数据交换
通过控制器中的数据寄存器来实现 CPU 与控制器之间、控制器与设备之间的数据交换。
（3）标识和报告设备的状态
通过控制器中的状态寄存器记下设备的状态供 CPU 了解。
（4）地址识别
通过控制器中的地址译码器来识别各个设备。系统中的每一个设备都有一个地址，用来唯一的标示这台设备，为此设备控制器必须能够识别它所控制的每个设备的地址。
（5）数据缓冲
通过控制器中的缓冲器来缓解 I/O 设备与 CPU 之间速率不匹配的矛盾。
（6）差错控制
设备控制器兼管数据的差错检查。通过将差错检测码置位来向 CPU 报告数据出错。
45、通道（是硬件）
概念：I/O 通道是一种特殊的处理机，它具有执行 I/O 指令的能力，并通过执行通道程序（I/O 程序）来控制 I/O 操作。通道是可以与 CPU 并行操作的专门用来控制输入输出设备进行数据传送的处理机。
特点：
①指令类型单一，主要局限于 I/O 操作有关的指令
②与 CPU 共享内存
类型：（根据信息交换方式的不同分）
（1）字节多路通道（数据传送是按字节交叉方式工作）
①有一个主通道。
②含有多个非分配型子通道。
③每个子通道通过一个控制器与一台低速的 I/O 设备相连。
④各子通道以时间片轮转方式按字节交叉使用主通道。
特点：可分时并行操作；传输率较低。
（2）数组选择通道（数据传输按成组方式进行工作，每次传输一批数据，主要用于连接高速 I/O 设备）
①有一个主通道。
②含有一个分配型子通道。
③子通道通过一控制器与一台高速 I/O 设备相连，一段时间内只能执行一个子通道程序。
特点：数据传输率较高；通道利用率低。
（3）数组多路通道（将数组选择通道的传输速率高和字节多路通道的分时并行操作的优点结合起来，形成的一种新的通道）
①有一个主通道。
②含有多个非分配型子通道。
③每个子通道通过控制器与一台高、中速的 I/O 设备相连，可同时并行向主通道传数据。
④各子通道以时间片轮转方式按成组方式使用主通道。
特点：能分时并行操作，数据传输率和通道利用率较高。
46、设备驱动程序的处理过程
①将抽象要求转换为具体要求
②对服务请求进行校验
③读出和检查设备的状态

④传送必要的参数
⑤启动 I/O 设备
47、4 种 I/O 控制方式
（1）程序 I/O 控制方式
特点：CPU 和外设只能串行工作，严重降低了系统的性能
流程图：

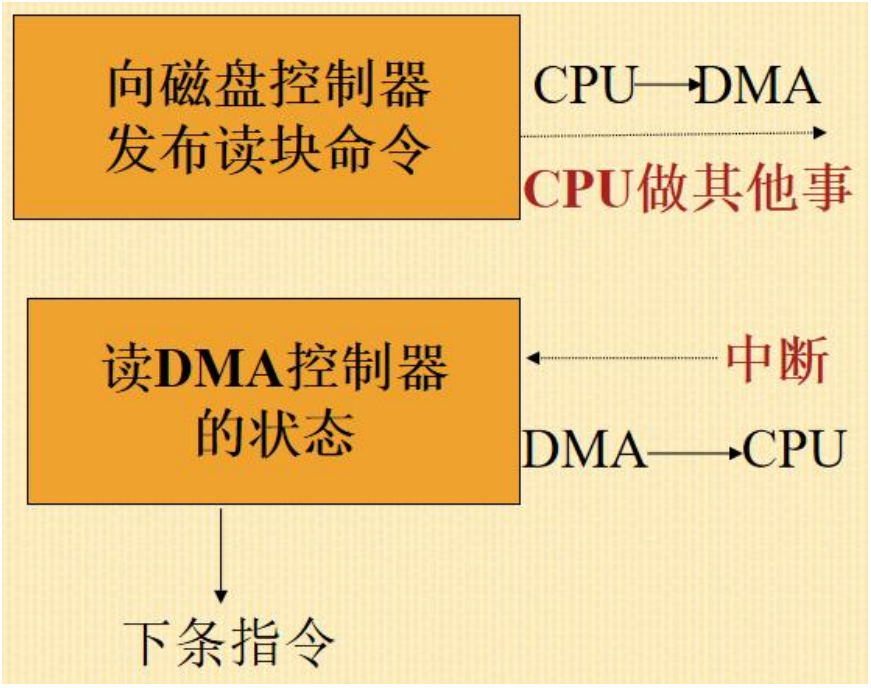


（2）中断驱动 I/O 控制方式
特点：CPU 与设备并行工作，传送信息的单位通常是一个字(节)，若中断次数较多，会耗去大量 CPU 的处理时间。提高了系统的资源利用率及吞吐量
流程图：

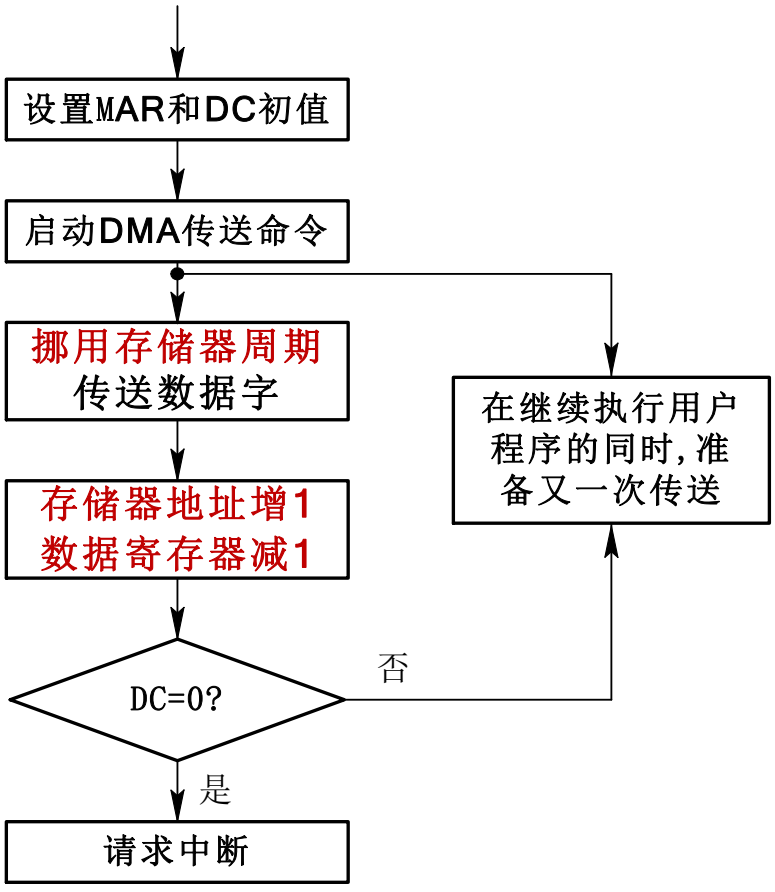


（3）直接存储器访问控制方式（DMA）

特点：数据传输基本单位为数据块，传输速率进一步提高；所传送数据从内存到设备，或相反；仅在传送一个连续数据块的开始和结束时才需中断 CPU。实现主机与控制器（外设）之间成块数据的直接交换，不用 CPU 插手（适用于高速的 I/O 设备）



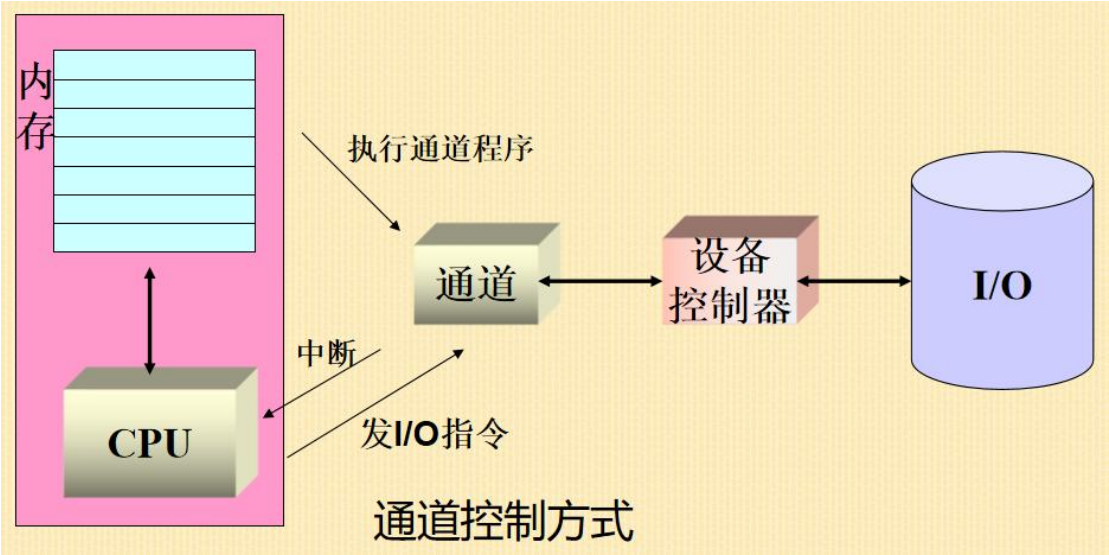
流程图：



（4）I/O 通道控制方式

特点：通道技术是对一组数据块的读（或写）及有关的控制和管理为单位的干预。从而实现 CPU、通道和 I/O 设备三者之间的并行工作。

图解：



48、什么是设备无关性？

用户程序不直接使用物理设备名（或设备的物理地址），而只能使用逻辑设备名；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。

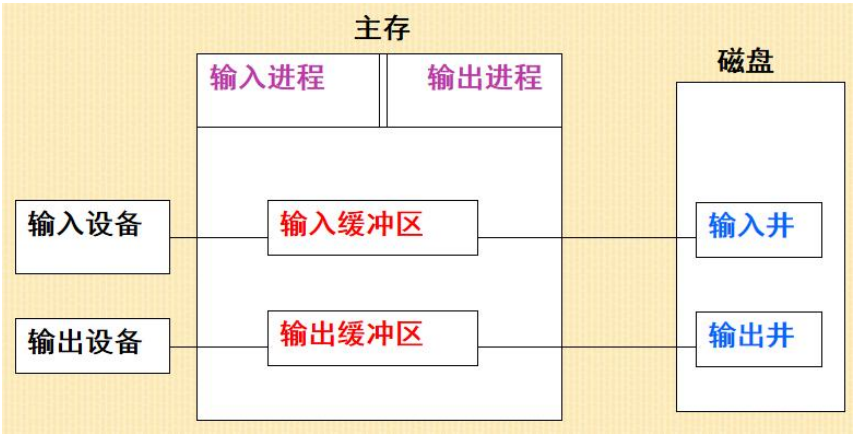
49、映射逻辑名和物理名

系统中设置逻辑设备表（LUT），用于实现将用户程序中所使用的逻辑设备名映射为物理设备名。

50、Spooling 技术（假脱机技术）

引入目的：为了缓和 CPU 的高速性与 I/O 设备低速性之间的矛盾，在联机情况下实现的同时外围操作，Spooling 技术是对脱机输入、输出工作的模拟，它必须建立在具有多道程序功能的操作系统上，需要有高速的、大容量的随机存储器（如磁盘）支持，是操作系统中采用的一项将独占设备改造成为共享设备的技术。

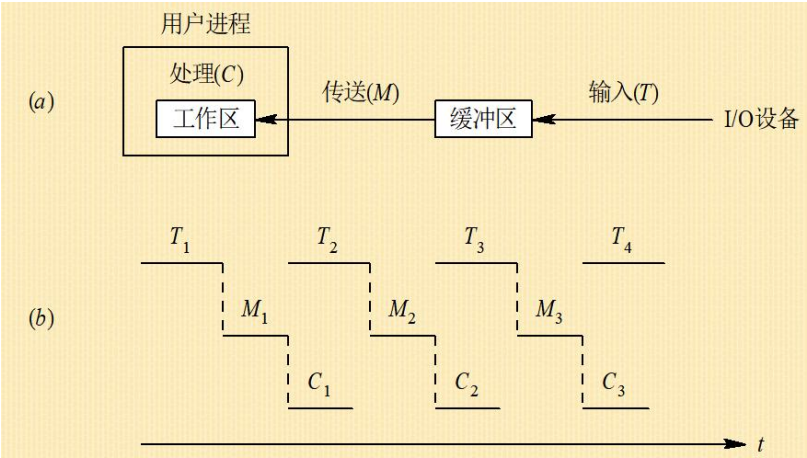
Spooling 技术的组成：



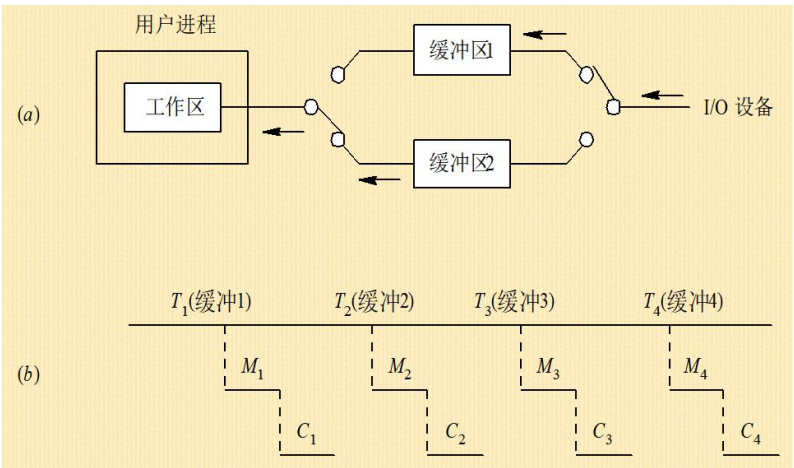
（1）输入井和输出井：在磁盘上开辟出来的两个存储区域

（2）输入缓冲区和输出缓冲区：在内存中开辟的两个缓冲区，用于缓和 CPU 和磁盘之间速度不匹配的矛盾。

（3）输入进程和输出进程
（4）井管理程序
特点：
（1）提高了 I/O 的速度。对低速 I/O 设备进行的 I/O 操作变为对输入井或输出井的操作。如同脱机操作一样，缓和了 CPU 与低速 I/O 设备间速度不匹配的矛盾。
（2）将独占设备改造为共享设备。
（3）实现了虚拟设备功能。
51、单缓冲和双缓冲
引入缓冲的原因：
（1）缓和 CPU 与 I/O 设备间速度不匹配的矛盾
（2）减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制
（3）解决数据粒度不匹配的问题
（4）提高 CPU 和 I/O 设备之间的并行性
软缓冲（在内存中）：单缓冲、双缓冲、循环缓冲、缓冲池
单缓冲：



在设备和处理机之间设置一个缓冲区。设备与处理机交换数据时，先把交换的数据写入缓冲区，然后需要数据的设备/处理机再从缓冲区中取走数据。
在单缓冲情况下，磁盘把数据输入到缓冲区的操作和 CPU 对数据的计算过程可以并行展开，系统对每一块数据的平均处理时间近似为 max(C, T) + M。
（注：在块设备输入时，先从磁盘把一块数据输入到缓冲区，耗时为 T；然后由操作系统将缓冲区数据传送给用户区，耗时为 M；接下来便由 CPU 对这一块数据进行计算，耗时 C。）
双缓冲：



在设备和处理机之间设置两个缓冲区。从设备输入数据时，先把数据装入第一缓冲区，装满后便转向第二缓冲区，此时 OS 可以从缓冲区 A 中提取数据传送到用户区，最后由 CPU 对数据进行计算。

将数据从缓冲区传送到用户区操作必须与 CPU 对数据处理串行进行，而且它们又可以与从外存传送数据填满缓冲区的操作并行，因此耗时大约为 max(C+M，T)。

联系：两台机器之间通信时，可以配置缓冲区用于数据的发送和接收。若两个相互通信的机器只设置单缓冲区，在任一时刻只能实现数据的单向传输。两个相互通信的机器设置双缓冲区，则同一时刻可以实现双向的数据传输。

52、文件的物理结构和逻辑结构

物理结构：

（1）顺序式文件（连续分配）

（2）链接式文件（链接分配）

①隐式：适合顺序存储

②显示

（3）索引式文件（索引分配）

逻辑结构：

按文件是否有结构分

（1）有结构文件（记录式文件）

（2）无结构文件（流式文件）

按文件的组织方式分（有结构文件）

（1）顺序文件

（2）索引文件

（3）索引顺序文件

选取文件逻辑结构时应遵循的原则：

（1）能减少修改文件时的处理工作量

（2）能有较快的查找速度

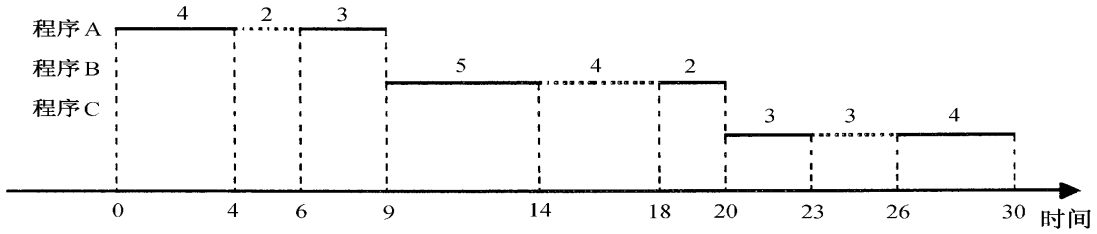
（3）能尽量节约存储空间

（4）便于用户进行操作

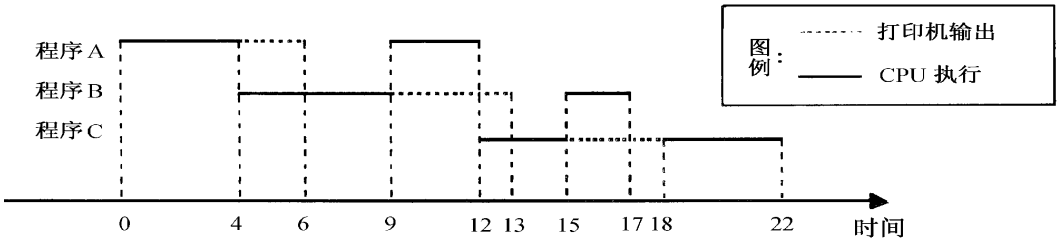
----------老师勾的重点到此结束----------

----------以下是本人自己总结的重点----------

1、求解 CPU 的利用率（分别在单道、多道程序环境下）



(a) 单道程序设计环境



(b) 多道程序设计环境

在单道程序环境下的 CPU 利用率为：$\frac{21}{30} = 0.7$

在多道程序环境下的 CPU 利用率为：$\frac{21}{22} \approx 0.95$

2、进程的创建、终止过程、阻塞过程、唤醒过程

进程的创建：

（1）申请一个空白的 PCB（进程控制块）

（2）为新进程分配其运行所需的资源

（3）初始化进程控制块（PCB）

（4）如果进程就绪队列能够接纳新进程，便将新进程插入就绪队列

进程的终止：

（1）根据终止的进程标识符，从 PCB 集合中检索出该 PCB，读出该进程的状态

（2）若处于执行状态，应立即终止进程执行，并置调度标志为真

（3）若还有子孙进程，将其所有子孙进程都要终止

（4）被终止的进程将全部资源归还父进程或系统

（5）将进程（PCB）从所在队列中移出

进程的阻塞：

（1）调用 block 原语将自己阻塞，停止执行

（2）修改 PCB 的状态，从执行态到阻塞态

（3）将 PCB 插入到阻塞队列

（4）切换 CPU 调度另一就绪程序

进程的唤醒：

（1）调用 wakeup 原语将等待该事件的进程唤醒，从阻塞队列移出

（2）修改 PCB 的状态，从阻塞态到就绪态

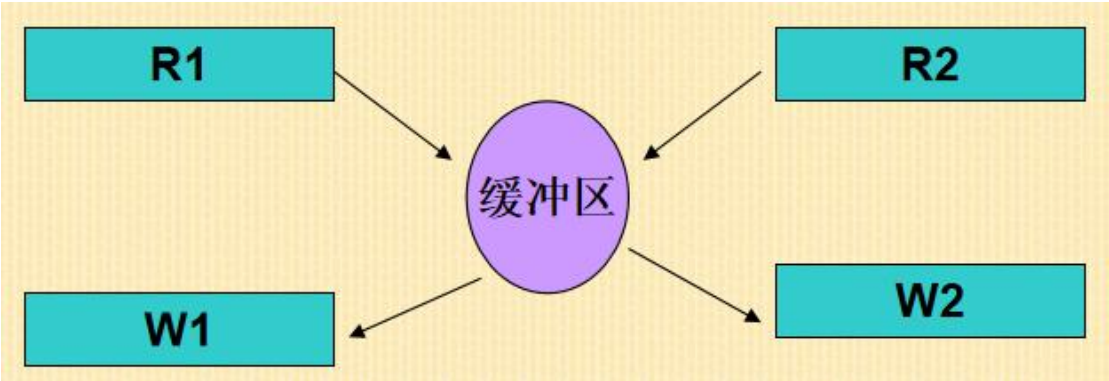
（3）再将该 PCB 插入到就绪队列中

3、信号量机制解决进程同步、进程互斥问题

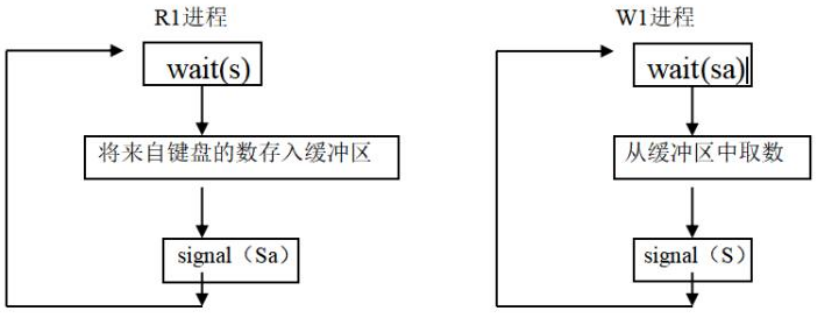
（解题思路：定义信号量，设置初始值，写算法）

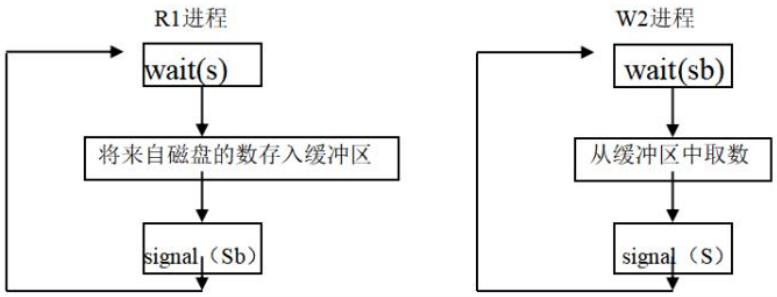
例题：现有四个进程 R1，R2，W1，W2，它们共享可以存放一个数的缓冲区。进程 R1 每次把来自键盘的一个数存入缓冲区中，供进程 W1 打印输出；进程 R2 每次从磁盘上读一个数存

放到缓冲区中，供进程 w2 打印输出。为防止数据的丢失和重复打印，问怎样用 wait、signal 操作来协调这四个进程的并发执行。



设置三个信号量 s、so、sa,信号量 s 表示空缓冲区，其初值为 1；信号量 sa 表示缓冲区中是否有来自键盘的数，其初值为 0；信号量 sb 表示缓冲区中是否有来自磁盘的数，其初值为 0。同步描述如下：





4、周转时间、带权周转时间

周转时间 = 作业完成时间 - 作业到达时间

带权周转时间 = 周转时间 / 服务时间

5、作业：用户提交的任务实体

作业的组成：程序、数据、作业说明书（因为人机不能交互所以才需要）

作业控制块（JCB）是作业在系统中存在的标志，保存了系统对作业进行管理和调度所需的全部信息。

作业运行的三个阶段、三个状态：

收容、运行、完成三个阶段；收容状态（后备状态）、运行状态、完成状态

6、先来先服务调度算法（FCFS）

按照作业到达的先后次序进行调度，有利于长作业（CPU 型），不利于短作业（I/O 型）

例题：

作业名	到达时间	服务	开始执行时间	完成时间	周转时间	带权
A	0	1	0	1	1	1
B	1	100	1	101	100	1
C	2	1	101	102	100	100
D	3	100	102	202	199	1.99

7、短作业优先调度算法（SJF）

以作业的长短来计算优先级，作业越短，优先级越高，有利于短作业（I/O 型），不利于长作业（CPU 型）。

优点：

（1）降低作业的平均等待时间

（2）提高吞吐量

（3）缩短周转时间

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）未考虑作业的紧迫程度，不能保证紧迫性任务得到及时处理

例题：

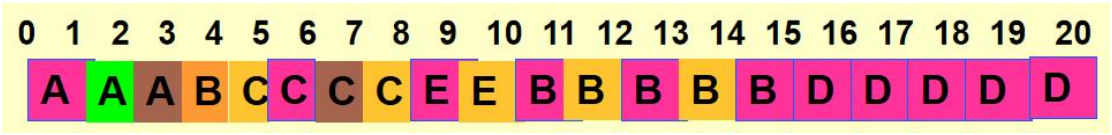
Problem1:应用最短作业优先（非抢占式）的作业调度算法，试将下面表格填写完整。

作业	进入系统时间	估计运行时间/分钟	开始时间	完成时间	周转时间/分钟
1	8:00	40	8:00	8:40	40
2	8:20	30	8:52	9:22	62
3	8:30	12	8:40	8:52	22
4	9:00	18	9:27	9:45	45
5	9:10	5	9:22	9:27	17
作业平均周转时间 T=37.2					

Problem2:假设一个系统中有 5 个作业，它们的到达时间和服务时间如表所示，忽略 I/O 以及其他开销时间，采用抢占的短作业优先（SJF）进行调度，请给平均周转时间。

作业	到达时间	服务时间
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

分析：



作业	到达时间	服务时间	完成时间	周转时间
A	0	3	3	3
B	2	6	15	13
C	4	4	8	4
D	6	5	20	14
E	8	2	10	2
平均周转时间 T=7. 2				

8、优先级调度算法（PSA）：分为抢占式和非抢占式。

9、高响应比优先调度算法（HRRN）

响应比（R_P）：$R_P = 1 + \frac{\text{等待时间}}{\text{服务时间}}$

特点：

（1）如果作业等待时间相同，则要求服务时间愈短，其优先权越高，类似 SJF，有利于短作业。

（2）如果作业服务时间相同，则要求等待时间愈短，优先权越高，类似 FCFS。

（3）对于长作业的优先级，可以随等待时间的增加而提高。

（4）每次都要计算响应比，增加了系统的开销，既要照顾长作业，又要照顾短作业。

例题：有四个作业，若采用响应比高者优先调度算法，求作业的调度顺序。

作业	到达时间	所需 CPU 时间
1	8.2	2
2	9.4	1
3	9.5	0.8
4	9.8	0.2

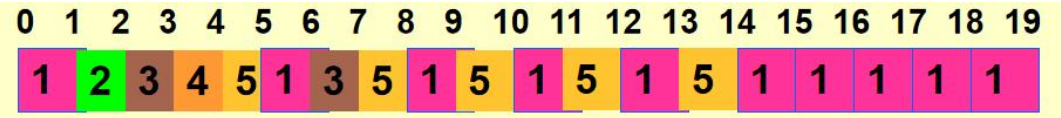
步骤（1）若采用响应比高者优先作业调度算法，先将作业 1 投入运行，于 10.2 完成，此时：

作 业	到达时间	所需 CPU 时间	已等待时间	响应比
2	9.4	1	0.8	1.8
3	9.5	0.8	0.7	1.875
4	9.8	0.2	0.4	3

步骤（2）由于作业 4 有最高响应比，因此被调度投入运行，于 10.4 完成，此时

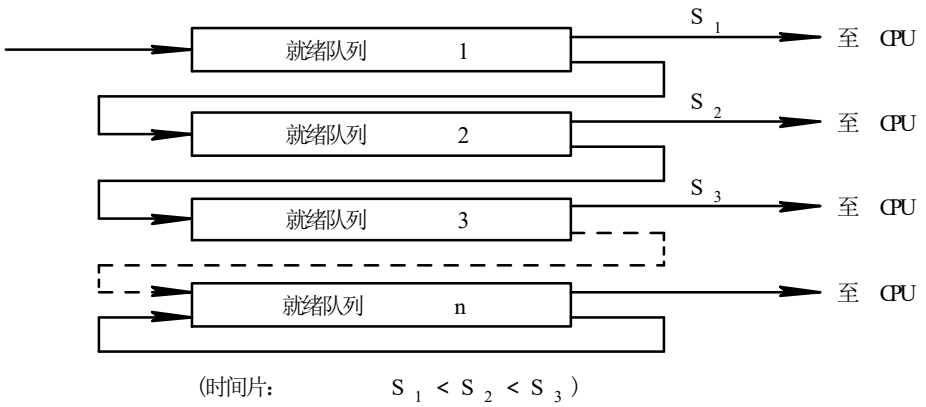
作 业	到达时间	所需 CPU 时间	已等待时间	响应比
2	9.4	1	1	2
3	9.5	0.8	0.9	2.125

步骤（3）由于作业 3 有最高响应比，因此被调度投入运行，于 11.2 完成。
步骤（4）作业 2 于 12.2 完成。
故调度顺序为：1、4、3、2
10、时间片轮转调度算法（RR）
特点：系统能在响应时间完成用户的需求。
基本思想：把 CPU 的处理时间分成固定大小的时间片，如果一个进程在调度中被选中后，用完系统规定的时间片任然没有完成任务，则让出处理机，并排到就绪队列末尾。同时进程调度程序又调度当前就绪队列对首的第一个进程投如运行。
切换进程的时机：
（1）若进程的执行时间大于时间片，调度程序将其送入就绪队列的末尾，并调度就绪队列的队首进程执行。
（2）若进程的执行时间少于时间片，调度程序立即调度就绪队列的队首进程，并启动一个新的时间片。
时间片 q 大小的确定：
q 过长：导致算法变成了先来先服务算法（FCFS）
q 过短：进程切换次数增加，导致系统开销过大
例题：假定在一台处理机上要运行以下作业 1,2,3,4,5.且假定这些作业在时刻 0 到达,他们的执行时间分别是:10、1、2、1、5。用时间片轮转法，当 q=1 时，各作业的周转时间和带权周转时间各是多少？



作业	服务时间	进入时刻	完成时刻	周转时间	带权
1	10	0	19	19	1.9
2	1	0	2	2	2
3	2	0	7	7	3.5
4	1	0	4	4	4
5	5	0	14	14	2.8

11、多级反馈队列调度算法（MFQ）
基本思想：多级反馈队列调度算法是时间片轮转算法和优先级调度算法的综合和发展，通过动态调整进程优先级和时间片大小，不必事先估计进程的执行时间，多级反馈队列可兼顾多方面的系统目标，是目前公认的一种较好的进程调度算法。
特点：较好的满足各种类型用户的需要



（1）设置多个就绪队列，并为每个队列赋予不同的优先级。队列 1 的优先级最高，其余队列逐个降低。

（2）每个队列中进程执行时间片的大小也各不相同，进程所在队列的优先级越高，其相应的时间片就越短。

（3）当一个新进程进入系统时，首先将它放入队列 1 的末尾，按 FCFS 等待调度。如在分配的时间片内能完成，便撤离系统，反之由调度程序将其转入队列 2 的末尾，按 FCFS 再次等待调度，如此下去，进入队列 n 按时间片轮转算法调度执行。

（4）仅当队列 1 为空时，才调度队列 2 中的进程运行。若队列 i 中的进程正执行，此时有新进程进入优先级较高的队列中，则新进程将抢占运行。正在运行的进程放回到第 i 队列的末尾。

12、优先级倒置现象

形成原因：高优先级进程（线程）被低优先级进程（线程）延迟或阻塞的现象，他们共享一个临界资源，低优先级的先使用，被高优先级的抢占 CPU，高优先级的也想使用临界资源，所以被阻塞了。

解决方法：采用动态优先级继承。

13、利用银行家算法避免死锁

数据结构：

（1）可利用资源向量 Available，Available[j]=K 表示系统中现有 j 类资源 K 个

（2）最大需求矩阵 Max，Max[i, j]=K 表示进程 i 需要 j 类资源的最大数目为 K 个

（3）分配矩阵 Allocation，Allocation[i, j]=K 表示进程 i 当前已分得 j 类资源的数目为 K 个

（4）需求矩阵 Need，Need[i, j]=K 表示进程 i 还需要 j 类资源 K 个方能完成任务

关系： $\mathrm{Need[i,j]} = \mathrm{Max[i,j]} - \mathrm{Allocation[i,j]}$

算法流程：

（1）合理性判断

（2）试探性分配

（3）安全性检测（安全性算法）

例题：假定系统中有五个进程｛P0, P1, P2, P3, P4｝和三类资源｛A, B, C｝，各种资源的数量分别为 10、5、9，在 T0 时刻的资源分配情况如图所示。P0 请求资源 Req(1,0,2)

进程＼资源	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	3	3	2
				1	1	2	6	4	1	2	3	0
P1	3	2	2	2	0	2	1	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

解题步骤：

（1）合理性判断

$$Req_0(1,0,2) \le Need_0(7,4,3)$$

$$Req_0(1,0,2) \le Available(3,3,2)$$

（2）试探性分配

（直接在题目写）

（3）安全性算法

	Work			Allocation			Available		
P1	2	3	0	2	0	2	4	3	2
P3	4	3	2	2	1	1	6	4	3
P0	6	4	3	1	1	2	7	5	5
P2	7	5	5	3	0	2	10	5	7
P4	10	5	7	0	0	2	10	5	9

对安全性算法后能找到安全序列 P1、P3、P0、P4、P2，所以能将资源分配给进程 P0 进程。

14、死锁的检测

资源分配图：S 为死锁的充分条件是，当且仅当 S 状态的资源分配图是不可完全简化的。该充分条件被称为死锁定理。（死锁一定有环，有环不一定死锁）

资源 -> 进程：分配边

进程 -> 资源：请求边

化简资源分配图的方法：

（1） 在资源分配图中，找出一个既不阻塞又非孤立的进程节点 Pi，如果找到，转（2），否则转（3）。

（2）消去 Pi 的分配边和请求边，使其变为孤立节点，转（1）

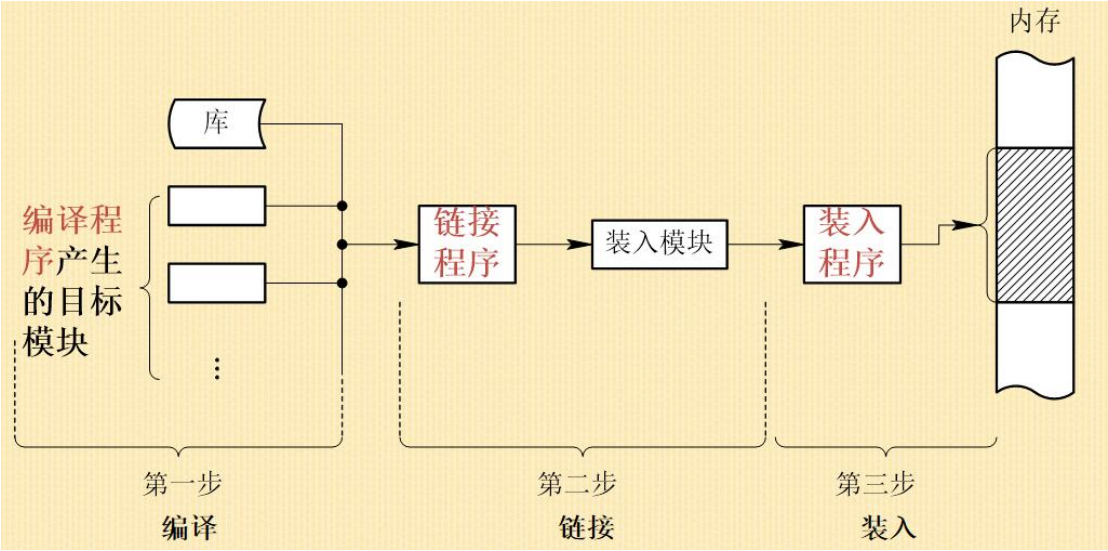
（3）若能消去资源分配图中所有的边，使所有进程成为孤立节点，则该图是可完全简化的，否则为不可简化。

15、用户程序变为可执行程序需要经过以下几个步骤：

（1）编译

（2）链接

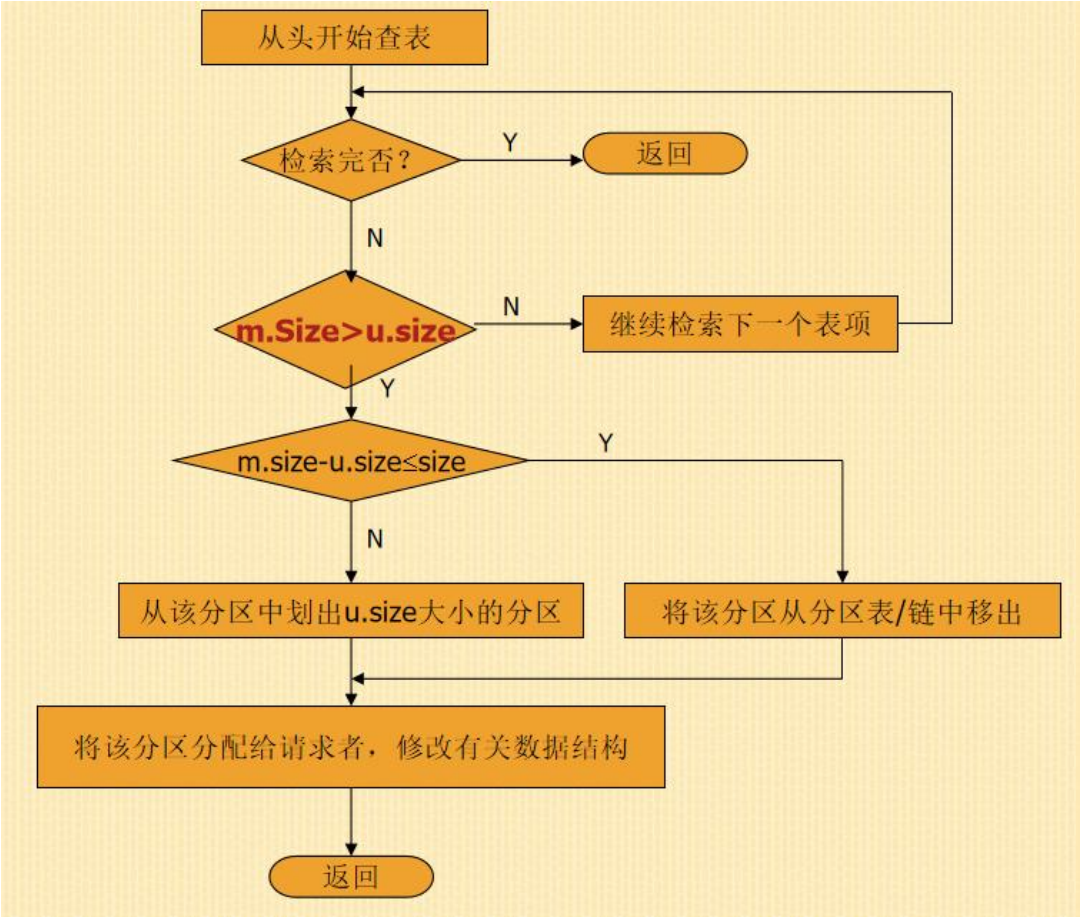
（3）装入



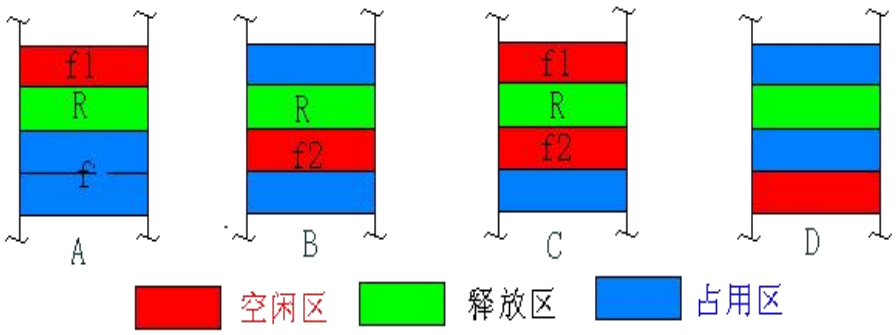
16、动态分区分配算法流程

设请求的分区大小为 u.size，空闲分区的大小为 m.size，若 m.size-u.size≤size(size 是事先规定的不再切割的剩余分区的大小)，说明多余部分大小，可不再切割，将整个分区分配给请求者；否则，从该分区中按请求的大小划分出一块内存空间分配出去，余下的部分仍留在空闲分区表/链中，然后，将分配区的首址返回给调用者。

分配流程图：



内存回收机制：



A（有前临空闲分区）：将 f1 与 R 合并，f1.addr；f1.size+R.size

B（有后临空闲分区）：将 R 与 f2 合并，R.addr；R.size+f2.size

C（有前后临空闲分区）：将 f1、R、f2 合并，f1.addr；f1.size+R.size+f2.size

D（无前后临空闲分区）：R 作为一个单独的空闲区

例题：页式存储管理允许用户用的编程空间为 32 个页面（每页 1KB），主存为 16KB，一个用户程序长 10 页，且某时刻该用户程序页表如图：计算逻辑地址 0AC5H 对应的物理地址

页号	块号
0	8
1	7
2	4
3	10

解：因页面大小为 1KB，所以逻辑地址低地址 10 位为页内偏移量，用户编程空间 32 个页面，逻辑地址高地址 5 位（因为$2^5 = 32$）为页号（逻辑地址总共 15 位）

主存 16KB（2^{10}页面大小 $\times$ 2^4物理块号），物理地址的高 4 位为物块号（物理地址总共 14 位）

逻辑地址 0AC5H 转为二进制是 000 1010 1100 0101B,

页号为 2，块号为 4，对应的物理地址为 01 0010 1100 0101B，即 12C5H

17、请求分页中的内存分配

（1）最小物理块数的确定

（2）内存分配策略

①固定分配局部置换：为每个进程分配固定数目的物理块，在整个运行中都不改变。如出现缺页，则从中置换一页，再调入一页，该进程的内存空间不变。

②可变分配局部置换：分配一定数目的物理块，若发现缺页，则从该进程的页面中置换一页，根据该进程缺页率高低，则可增加或减少物理块。

③可变分配全局置换：分配一定数目的物理块，但 OS 自留一空闲块队列，若发现缺页，则从空闲块队列中分配一空闲块与该进程，并调入缺面于其中。当空闲块队列用完时，OS 才从内存中任选择一页置换。

（3）物理块分配方法

①平均分配算法：平均分配给各个进程。

②按比例分配算法：根据进程的大小按比例分配给各个进程。

③考虑优先权的分配算法：将系统提供的物理块一部分根据进程大小先按比例分配给各个进程，另一部分再根据各进程的优先权适当增加物理块数。

18、页面置换算法（求缺页率）

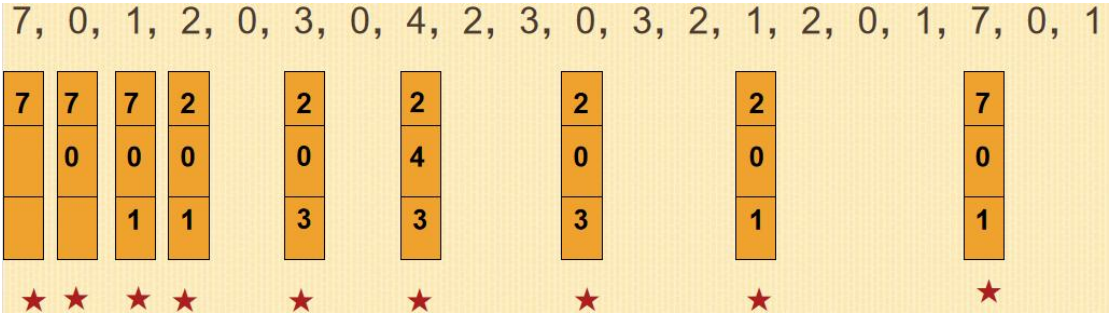
（1）最佳置换算法（理想化算法）

算法思想：选择将来不再被使用，或者最长时间不再被访问的页面将其淘汰。

算法特点：最佳置换算法产生的缺页中断次数最少，可获得最低的缺页中断率。但是由于无法预知哪个页面是未来最长时间内不被访问的，所以该算法只是一种理论上的算法，可用该算法评价其它算法的优劣。

例题：假设系统为某进程分配了三个物理块，程序访问页面的顺序为 7，0，1，2，0，3，0，4，2，3，0，3，2，1，2，0，1，7，0，1

使用最佳页面置换算法：



缺页次数为 9 次，缺页率为$\frac{9}{20}$

（2）先进先出置换算法

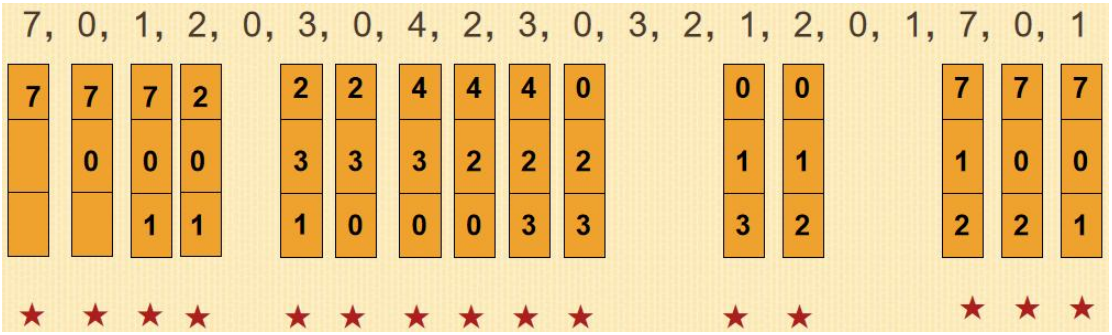
算法思想：选择最先进入内存的页面将其淘汰。

算法特点：先进先出(FIFO)置换算法实现简单，但与进程实际运行的规律不适应。

例题：假设系统为某进程分配了三个物理块，程序访问页面的顺序为 7，0，1，2，0，3，0，

4，2，3，0，3，2，1，2，0，1，7，0，1

使用先进先出置换算法：



缺页次数 15 次，缺页率为$\frac{15}{20}$

（3）最近最久未使用（LRU）置换算法

算法思想：选择最近最久没有使用的页面将其淘汰。该算法的出发点是如果某个页面被访问了，则它可能马上还要访问。反之，如果很长时间未被访问，则它在最近一段时间也不会被访问。

算法特点：最近最久未使用置换算法（LRU）的性能接近于最佳算法，但实现起来较困难。因为要快速找出最近最久未使用的页面，需要两类硬件之一的支持：寄存器或栈。

LRU 置换算法的硬件支持：

①寄存器

为每个在内存中的页面设立一个移位寄存器，以记录页面的访问情况。每当进程访问某页面时，将该页面对应寄存器的最高位置 1，系统定期将寄存器右移一位并将最高位补 0，寄存器数值最小的页面是最近最久未使用的页面。

②栈

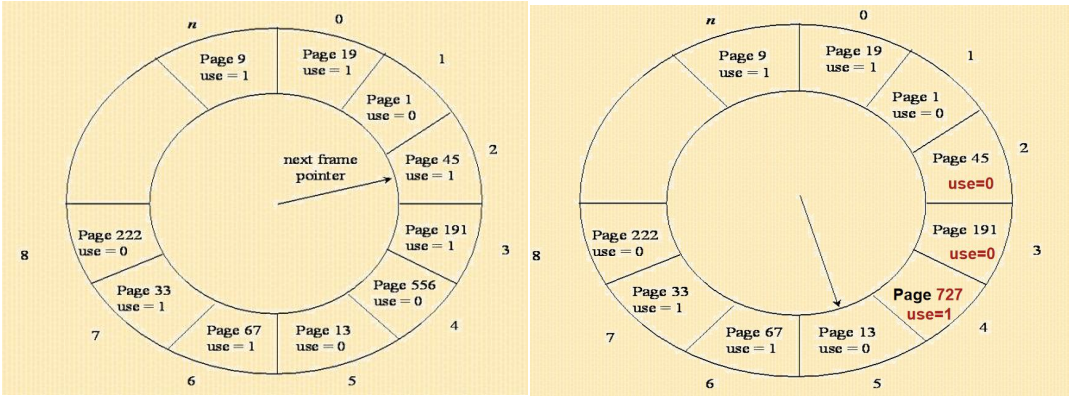
利用一特殊的栈保存当前使用的页号，每当进程访问某页面时，把被访问页面移到栈顶，于是栈底的页面就是最近最久未使用的页面。

（4）Clock 置换算法

A. 简单的 Clock 置换算法

算法思想：该算法要求为每页设置一个访问位，并将内存中的所有页链接成一个循环队列。当某页被访问时，系统将其访问位设置为 1。置换时采用一个指针，从当前指针位置开始按序检查各页，若访问位为 0 则选择该页换出，若访问位为 1 则将其设置为 0，最后指针停留在被置换页的下一页上。

假设需要访问处于外存上的第 727 页，需要置换出一页。



B. 改进型 Clock 置换算法

系统为内存的每个页面配置一个修改位(简称 M 位）。

• 改进后的 CLOCK 算法在选择被置换页面时将同时考虑访问位（A 位）和修改位（M 位）。

• 一个页面的 A 位与 M 位共有四种组合：

- A=0，M=0：最近没有被使用过，也没有被修改过；
- A=0，M=1：最近没有被使用过，但被修改过
- A=1，M=0：最近被使用过，但没有被修改过；
- A=1，M=1：最近被使用过，也被修改过；

在选择页面换出时，将最近既未使用过的（A=0）又未被修改过的（M=0）页面做为首选淘汰页面。

19、影响缺页率的因素

（1）分配给进程的物理块数

Belady 现象：虚拟存储系统中的一种异常现象，即增加进程的物理块数，缺页率反而上升。

（2）页面本身的大小

（3）程序的编制方法

（4）页面置换算法

20、CPU 利用率比较低的原因

（1）进程较少

（2）缺页率较高，产生抖动现象（进程太多，分配给每一个进程的物理块太少）

21、请求分段中的硬件支持

（1）请求段表机制

段名	段长	段的基址	存取方式	访问字段	修改位	存在位 P	增补位	外存地址

①存取方式：存取属性（执行、只读、允许读/写）

②访问字段 A：记录该段被访问的次数

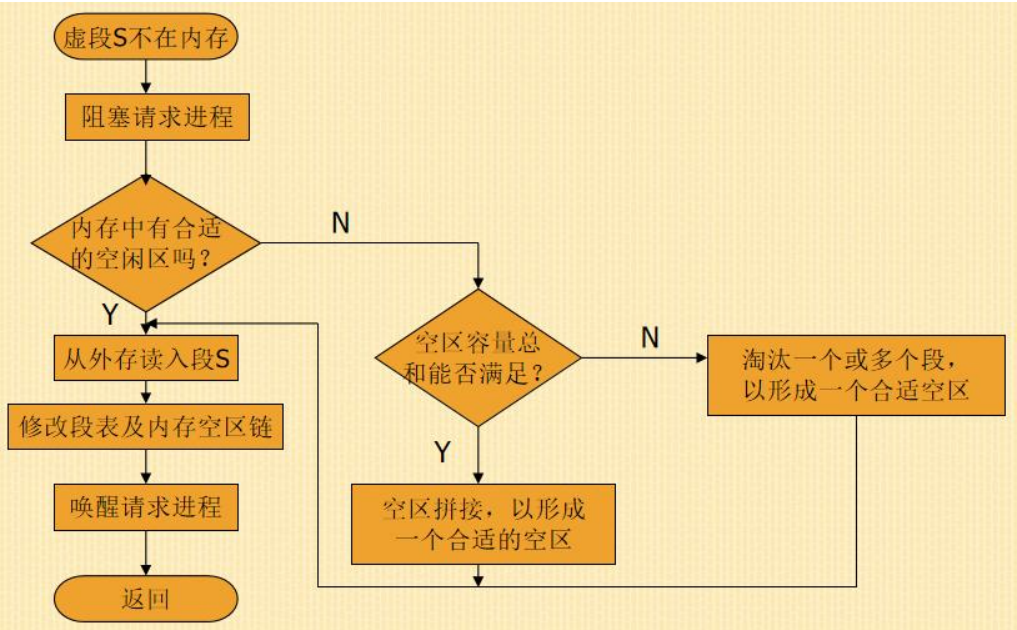
③修改位 M：表示该段在进入内存后，是否被修改过。

④存在位 P：表示该段是否在内存中。

⑤增补位：表示在运行过程中，该段是否做过动态增长。

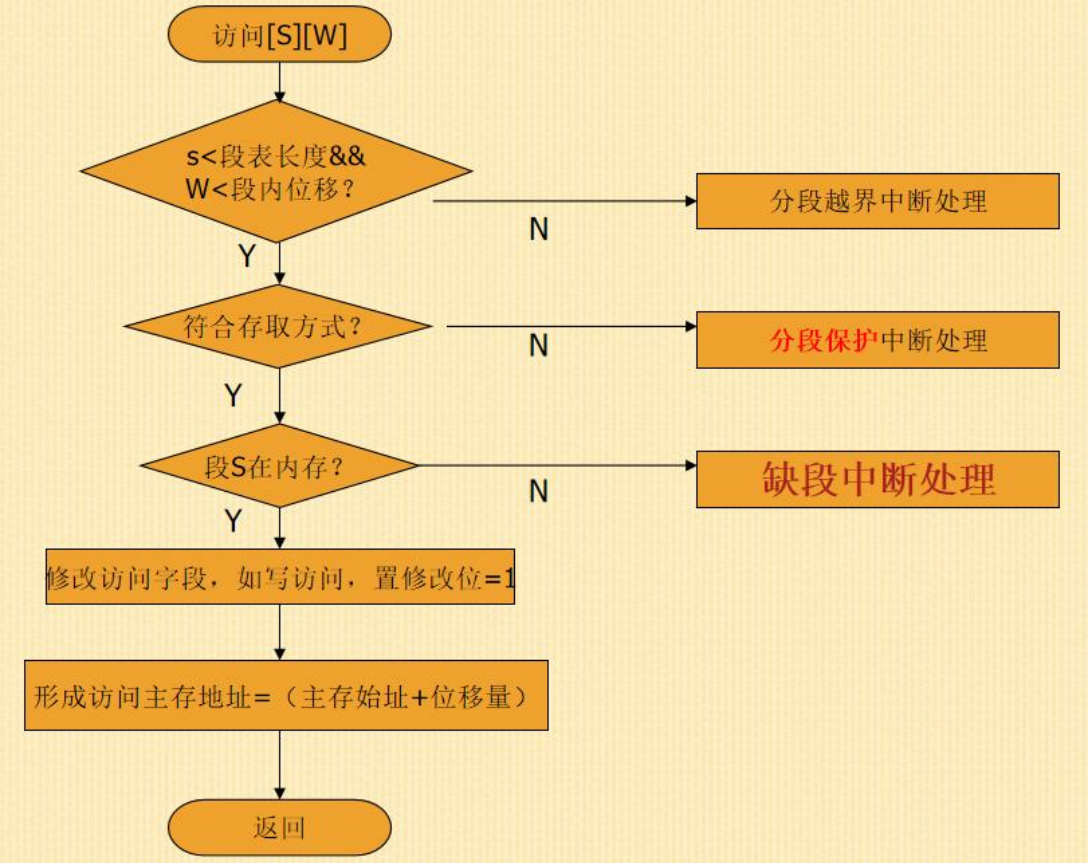
⑥外存地址：表示该段在外存中的起始地址。

（2）缺段中断机构



当被访问的段不在内存中时，将产生一缺段中断信号，由缺段中断程序将所需的段调入内存。

（3）地址变换机构



22、分段保护

（1）越界检查

先利用段表寄存器中的段表长度与逻辑地址中的段号比较，若段号超界则产生越界中断；再利用段表项中的段长与逻辑地址中的段内位移进行比较，若段内位移大于段长，也会产生越界中断。

（2）存取控制检查

在段表中设置了一个存取控制字段，用于规定对该段的访问方式。

（3）环保护机构

在环系统中，程序的访问和调用应遵循一定的规则：

①一个程序可以访问同环或较低特权环中的数据；

②一个程序可以调用同环或较高特权环中的服务；

23、按传输速率分类 I/O 设备

（1）低速设备：键盘、鼠标、语音的输入输出

（2）中速设备：打印机

（3）高速设备：磁盘、磁带、光盘

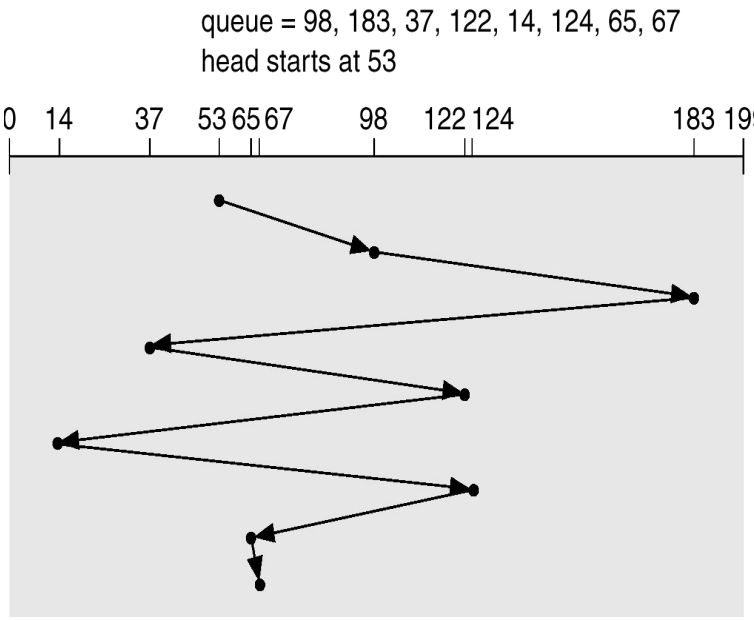
24、磁盘调度算法

（1）先来先服务 FCFS

基本思想：根据进程请求访问磁盘的先后次序进行调度。

特点：简单、公平，但平均寻道时间长。

例题：若干个等待访问磁盘者依次要访问的磁道为 98，183，37，122，14，124，65，67，移动臂当前位于 53 号磁道，则 FCFS 的平均寻道长度为多少？



下磁道	移道数
98	45
183	85
37	146
122	85
14	108
124	110
65	59
67	2
总道数	640
平均寻道长度	80

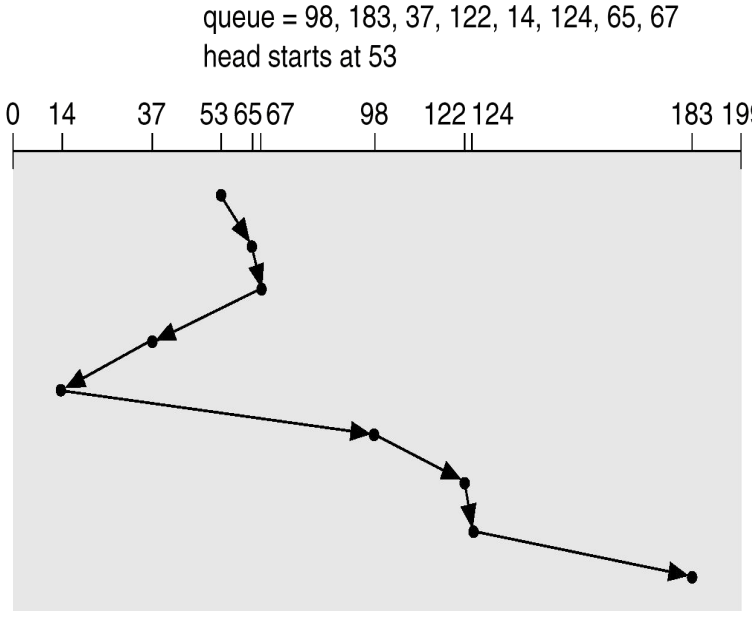
则 FCFS 的平均寻道长度为：80

（2）最短寻道时间优先 SSTF

基本思想：优先选择距当前磁头最近的访问请求进行服务。

特点：改善了磁盘平均服务时间，但容易造成某些访问请求长期等待得不到服务。

例题：若干个等待访问磁盘者依次要访问的磁道为 98，183，37，122，14，124，65，67，移动臂当前位于 53 号磁道，则 SSTF 的平均寻道长度为多少？



下磁道	移道数
65	12
67	2
37	30
14	23
98	84
122	24
124	2
183	59
总道数	236
平均寻道长度	29.5

则 SSTF 的平均寻道长度为：29.5

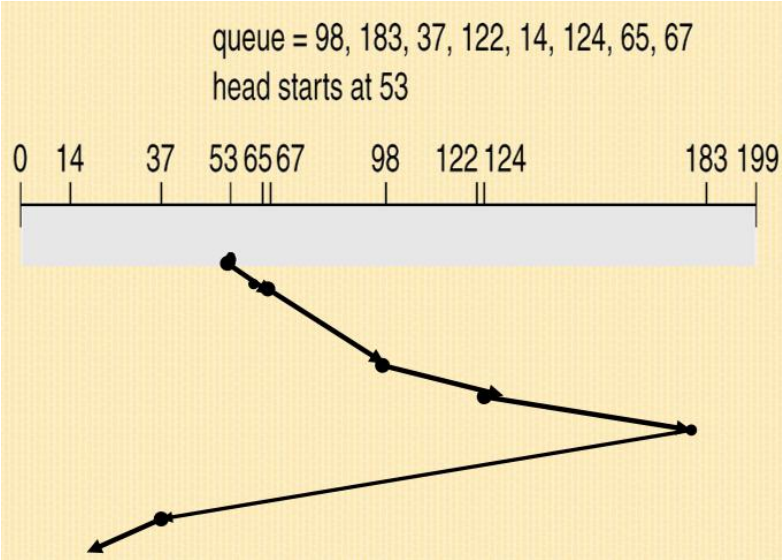
（3）扫描算法 SCAN

基本思想：磁头从磁盘的一端开始向另一端移动，沿途响应访问请求，直到到达了磁盘的另一端，此时磁头反向移动并继续响应服务请求。

特点：寻道性能较好，避免了饥饿，但不利于远离磁头一端的访问请求。

例题：若干个等待访问磁盘者依次要访问的磁道为 98，183，37，122，14，124，65，

67, 移动臂当前位于 53 号磁道，则 SCAN 算法（向磁道号增加的方向开始访问）的平均寻道长度为多少？



下磁道	移道数
65	12
67	2
98	31
122	24
124	2
183	59
37	146
14	23
总道数	299
平均寻道长度	37.4

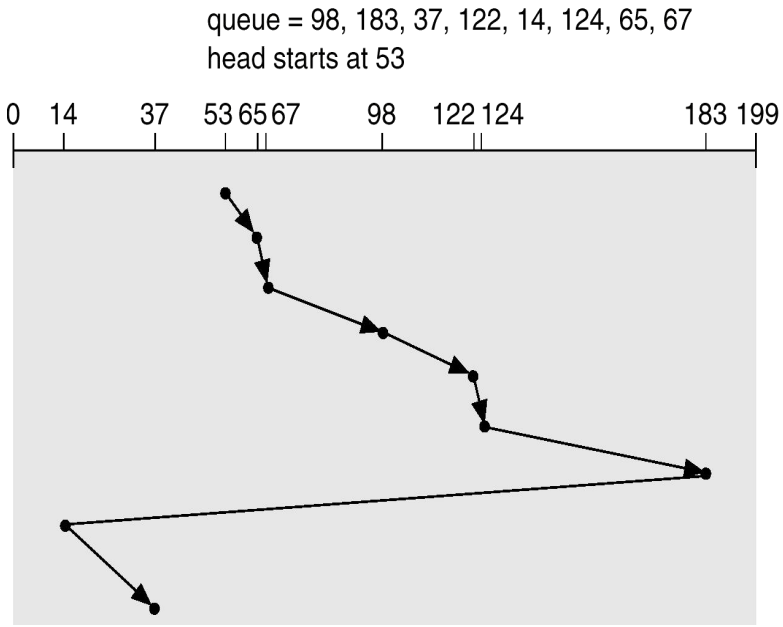
则 SCAN 算法的平均寻道长度为：37.4

（4）循环扫描算法 CSCAN

基本思想：类似于扫描算法，但磁头只能单向移动

特点：消除了对两端磁道请求的不公平。

例题：若干个等待访问磁盘者依次要访问的磁道为 98，183，37，122，14，124，65，67, 移动臂当前位于 53 号磁道，则 CSCAN 算法（向磁道号增加的方向开始访问）的平均寻道长度为多少？



下磁道	移道数
65	12
67	2
98	31
122	24
124	2
183	59
14	169
37	23
总道数	322
平均寻道长度	40.3

则 CSCAN 算法的平均寻道长度为：40.3

25、提高磁盘 I/O 速度的主要途径

（1）选择性能好的磁盘

（2）采用好的磁盘调度算法

（3）设置磁盘高速缓存（Disk Cache）
（4）其它方法