操作系统复习笔记

题目：

- 单选 3'*10
- 问答 10' * 7

问答 8 抽 7

- [x] 进程同步
- [x] 作业调度与进程调度
- [x] 银行家算法
- [x] 伙伴系统
- [x] 分页系统分段系统的地址转换
- [x] 页面置换
- [x] 磁盘调度
- [x] 实时调度

1. 引论

1.1 操作系统的目标

1. 方便性
2. 有效性: 提高系统资源的利用率和吞吐量
3. 可扩充性：微内核结构，方便地增添新功能和模块，以及对原有功能和模块进行修改
4. 开放性

1.2 操作系统的作用

1. 人机交互：OS 作为 用户与计算机硬件系统之间的接口
    1. 命令方式
    2. 系统调用方式
    3. 图形/窗口方式
2. 资源管理：OS 作为计算机系统资源的管理者
    1. 处理机：负责处理机的分配与控制
    2. 存储器：负责内存的分配与回收
    3. IO 设备：负责 IO 设备的分配与操纵
    4. 文件管理：实现对文件的存取、共享、保护
3. 资源抽象：：OS 实现了对计算机资源的抽象

1.3 操作系统发展的动力

1. 不断提高计算机系统的资源利用率
2. 方便用户
3. 器件不断更新迭代
4. 计算机体系结构不断发展
5. 不断提出新的应用需求

1.4 操作系统发展的过程

1.4.1 单道批处理系统

缺点：资源得不到充分利用

1.4.2 多道批处理

优点：

- 资源利用率高
- 系统吞吐量大

缺点：

- 平均周转时间长
- 无交互能力

周转时间：指一项作业提交给系统到执行结束的时间

1.4.3 操作系统的定义

OS 是一组能游戏笑傲地组织和管理计算机硬件和软件资源，合理地对各类作业进行调度，以及方便用户使用的程序的集合。

1.4.4 分时系统

能够 **人机交互** 和 **共享主机**

特征：

- 多路型
- 独立性
- 及时性
- 交互性

1.4.5 实时系统

及时完成某些作业

截止时间：

- 开始截止时间：某任务在某时刻前必须开始执行
- 完成结束及时：某任务在某时刻前必须结束执行

硬实时：必须满足限定时间

软实时：可以偶尔不满足限定时间

1.4.6 微机操作系统

1. 单用户单任务 OS：MS-DOS
2. 单用户多任务 OS：win98
3. 多用户多任务 OS：UNIX、类 UNIX、Windows NT

1.4.7 嵌入式操作系统

特点：

- 系统内核小
- 系统精简
- 实时性高
- 具有可配置性

1.4.8 网络操作系统

用于对网络资源进行管理和控制、实现数据通信及对网络资源的共享，为用户提供网络资源接口的彝族软件和规程的集合

1.4.9 分布式操作系统

1.5 操作系统的基本特性

1.5.1 并发

提高利用率，增加系统吞吐量

- 并发：多个事件在同一时间间隔内发生
- 并行：多个事件在同一时刻发生

引入进程实现并发

1.5.2 共享

指系统中的资源可供内存中多个并发执行的进程共同使用。

- 互斥共享：在一段时间内之允许一个进程访问的资源，称为临界资源，只能被互斥地访问
- 同时共享：宏观上可以有多个进程同时访问，微观上交替进行。

1.5.3 虚拟

时分复用、空分复用

1.5.4 异步

程序以人们不可预知的速度向前推进

1.6 操作系统内核

- 支撑功能：中断处理、时钟管理、原语操作
- 资源管理功能：进程管理、存储器管理、设备管理
- 双重工作模式：内核态与用户态

1.7 操作系统的主要功能

1.7.1 处理机管理

1. 进程控制
2. 进程 同步
3. 进程通信
4. 调度：作业调度（分配资源，调度进内存的就绪队列）、进程调度（从就绪队列中调度进处理机执行）

1.7.2 存储器管理

1. 内存分配与回收
2. 内存保护
3. 地址映射
4. 内存扩充

1.8 操作系统结构

1. 模块化：低耦合高内聚
2. 分层式：系统效率低
3. 微内核：基于 C/S 模式

2. 调度算法

2.1 作业调度

2.1.1 先来先服务（FCFS）

2.1.2 短作业优先（SJF）

2.1.3 高响应比优先（HRRN）

$$优先级 = \frac{等待时间+要求服务时间}{要求服务时间}$$

2.2 进程调度

2.2.1 轮转法

按先来先服务的顺序分配时间片

2.2.2 优先级算法

- 静态优先级
- 动态优先级
    - 同作业调度算法

2.2.3 多级反馈队列算法

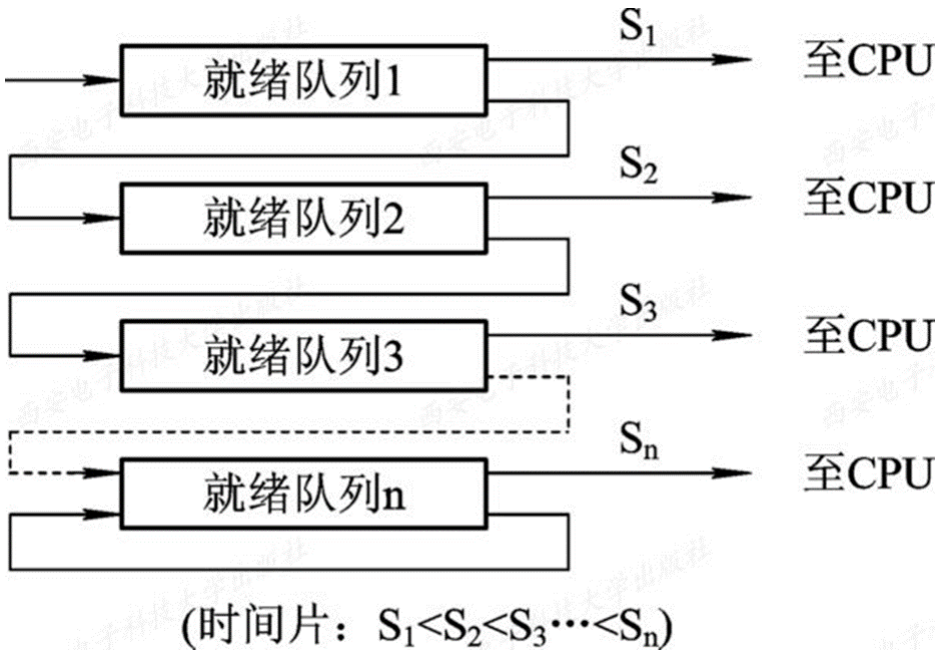


Figure 1

每个队列内按先来先服务调度

队列间用优先级调度

2.2.4 基于公平原则的调度算法

2.2.4.1 保证调度

按照进程数量均分

2.3 实时调度

2.3.1 分类

任务类型：硬实时和软实时

抢占类型：抢占式和非抢占式

抢占式又可分为基于时间片抢占和立即抢占

2.3.2 最早截止时间优先 EDF

2.3.2.1 非抢占式用于非周期任务

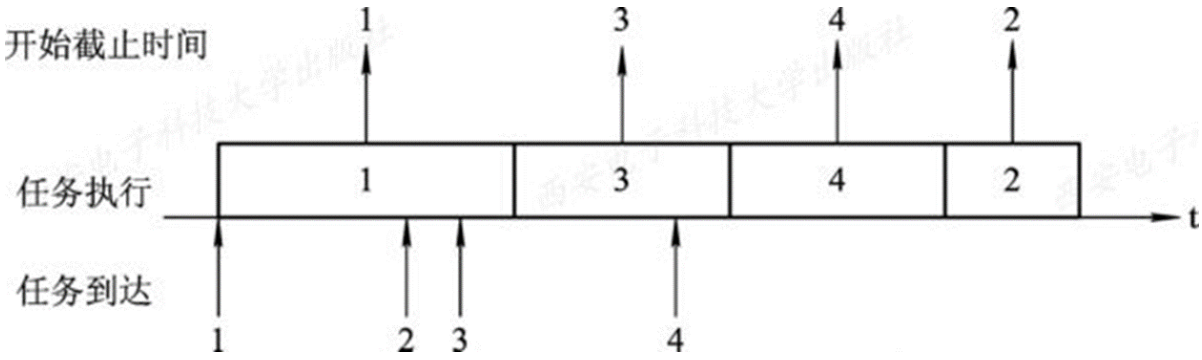


Figure 2

2.3.2.2 抢占式用于周期任务

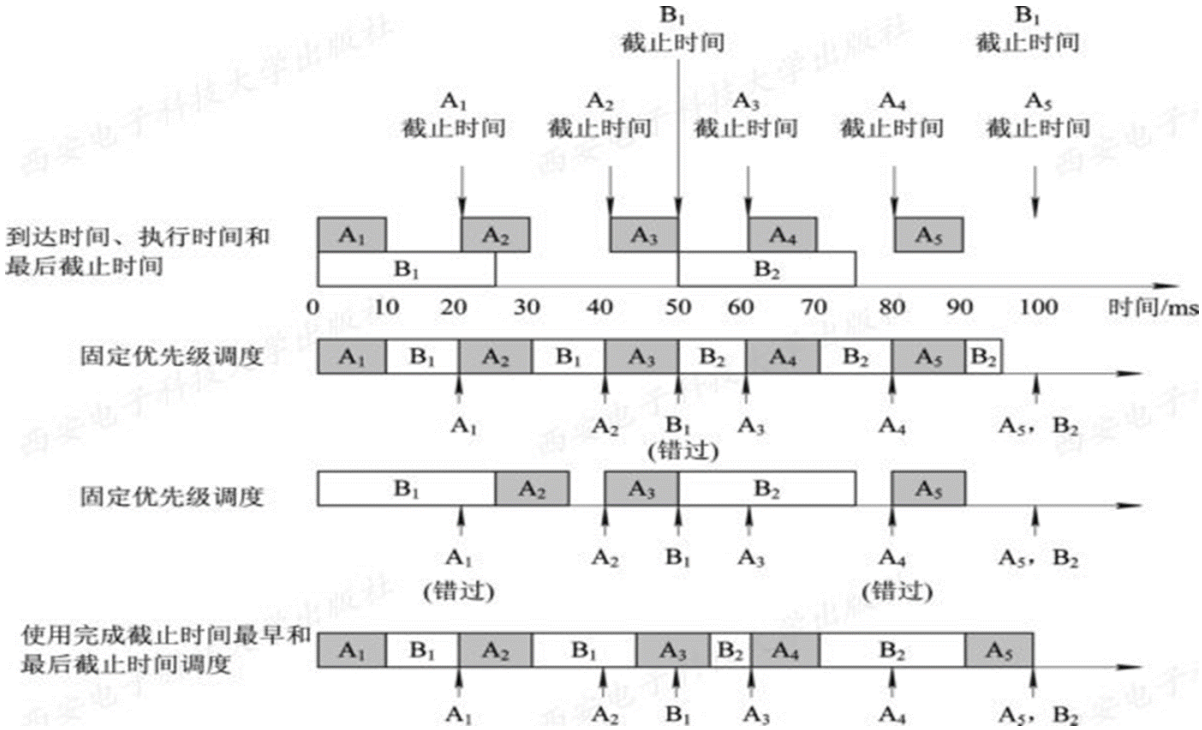


Figure 3

2.3.3 最低松弛度优先LLF

主要由于抢占式

松弛度 = 必须完成时间 − 剩余未运行时长 − 当前时间

松弛度越低，优先级越高；松弛度为 0 时需立即抢占

2.4 死锁

死锁的必要条件：

- 互斥条件
- 请求和保持条件
- 不可抢占条件
- 循环等待条件

处理死锁的方法

- 预防
- 避免

- 检测
- 解除

2.4.1 银行家算法

为了实现银行家算法，在系统中必须设置这样四个数据结构，分别用来描述系统中可利用的资源、所有进程对资源的最大需求、系统中的资源分配，以及所有进程还需要多少资源的情况。

(1) 可利用资源向量 Available。

(2) 最大需求矩阵 Max。

(3) 分配矩阵 Allocation。

(4) 需求矩阵 Need。

设 Requesti 是进程 Pi 的请求向量，如果 Requesti [j] = K，表示进程 Pi 需要 K 个 Rj 类型的资源。当 Pi 发出资源请求后，系统按下述步骤进行检查：

(1) 如果 Requesti [j] ≤Need [i, j]，便转向步骤(2)；否则认为出错，因为它所需要的资源数已超过它所宣布的最大值。

(2) 如果 Requesti [j] ≤Available [j]，便转向步骤(3)； 否则，表示尚无足够资源，Pi 须等待。

(3) 系统试探着把资源分配给进程 Pi，并修改下面数据结构中的数值：

Available [j] = Available [j] - Requesti [j];

Allocation [i, j] = Allocation [i, j] + Requesti [j];

Need [i, j] = Need [i, j] - Requesti [j];

(4) 系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。若安全，才 **正式** 将资源分配给进程 Pi，以完成本次分配；否则，将本次的试探分配作废，恢复原来的资源分配状态，让进程 Pi 等待。

2.4.2 银行家安全性算法

(1) 设置两个向量：

① 工作向量 Work，它表示系统可提供给进程继续运行所需的各类资源数目，它含有 m 个元素，在执行安全算法开始时，Work:= Available；

② Finish：它表示系统是否有足够的资源分配给进程，使之运行完成。

开始时先做 Finish [i]:= false；

当有足够资源分配给进程时，再令 Finish [i]:= true。

(2) 从进程集合中找到一个能满足下述条件的进程：

① Finish [i] = false;

② Need [i, j] ≤Work [j];

若找到，执行步骤(3)，否则，执行步骤(4)。

(3) 当进程 Pi 获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：

Work [j] = Work [j]+Allocation [i, j];

Finish [i] = true;

go to step 2;

(4) 如果所有进程的 Finish [i] = true 都满足，则表示系统处于安全状态；否则，系统处于不安全状态。

3. 存储器管理

3.1 伙伴系统

在伙伴系统中，对于一个大小为 2^k，地址为 x 的内存块，其伙伴块的地址则用 buddyk(x)表示，其通式为：

$$\mathrm{buddy}_k(x)=\begin{cases} x+2^k\ (x \bmod 2^{k+1}=0) \\ x-2^k\ (x \bmod 2^{k+1}=2^k) \end{cases}$$

Figure 4

3.2 内存有效访问时间EAT

一般情况下，处理机需要现在内存中访问页表，获得页号和页内地址，在处理机中计算出物理地址，这样需要访问两边内存，非常耗时。t，内存访问时间。

$$EAT = t + t$$

我们引入快表缓冲器和一个寄存器，使得页表中的地址直接送给寄存器，算出物理地址，存入快表中。

快表命中率为a，访问时间为 λ

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

可以显著提高。

3.3 分页系统与分段系统

3.3.1 分页系统

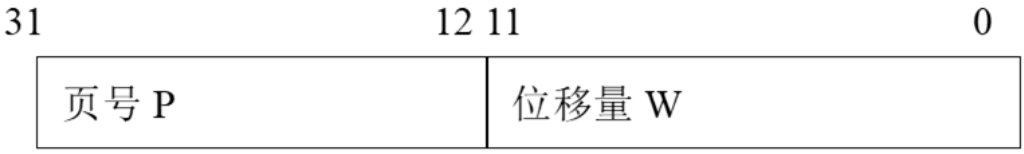


Figure 5

对某特定机器，其地址结构是一定的。若给定一个逻辑地址空间中的地址为 A，页面的大小为 L，则页号 P 和页内地址 d 可按下式求得：

$$P = INT[\frac{A}{L}], d = [A] \mod L$$

3.3.1.1 页表

1. 页表项数=程序的分页数目=页号+1=逻辑地址/页表大小
2. 页表项长度*页面的个数=页表长度=页表大小
3. 页表项中的内容：物理块号

例题：某系统采用页式存储管理策略，拥有逻辑空间32页，每页2KB，拥有物理空间1MB。请回答以下问题：
(1)画出逻辑地址的格式。
(2)若不考虑访问权限等，进程的页表有多少项？每项至少有多少位？

解：(1) 该系统拥有逻辑空间32页，故逻辑地址中页号必须用5位来描述；而每页为2KB，因此，页内地址必须用11位来描述。

(2) 每个进程最多有32个页面。因此，进程的页表项最多为32项；若不考虑访问权限等，则页表项中只需给出页所对应的物理块块号，1MB的物理空间可分为2^9个内存块，故每个页表项至少有9位。

3.3.1.2 多级页表

例6：已知某系统页面长4KB，每个页表项4B，采用多层分页策略映射64位的用户地址空间。若限定最高层页表只占1页，问它可采用几层分页策略？解：该系统的用户地址空间为2^{64}B，而页的大小为4KB，故一作业最多可有$2^{64}/2^{12}$(即2^{52})个页，其页表的大小则为$2^{52}*4$(即2^{54})B。因此，又可将页表分成2^{42}个页表项，并为它建立两级页表，两级页表的大小为2^{44}B。

依次类推，可知道它的3、4、5、6级页表的大小(长度)分别为2^{34}B、2^{24}B、2^{14}B、2^4B，故可采取6层分页策略。

3.3.2 分段系统

在分段存储管理方式中，作业的地址空间被划分为若干个段，每个段定义了一组逻辑信息。例如，有主程序段MAIN、子程序段X、数据段D及栈段S等，如图4-19所示。

分段地址中的地址具有如下结构：

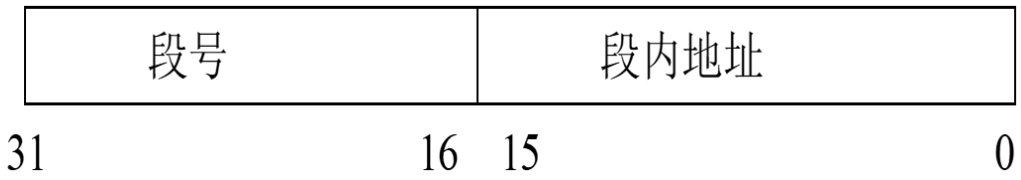


Figure 6

每个段都从0开始编址，并采用一段连续的地址空间。段的长度由相应的逻辑信息组的长度决定。因此，各段的长度并不相等。

3.3.2.1 段表

为此，在系统中，需为每个进程建立一张段映射表，简称“段表”。每个段在表中占有一个表项，其中记录了该段在内存中的起始地址(基址)和段的长度。

为了实现进程从逻辑地址到物理地址的变换功能，在系统中设置了段表寄存器，用于存放段表始址和段表长度TL。在进行地址变换时，系统将逻辑地址中的段号S与段表长度TL进行比较。
(1)若S>=TL，表示段号太大，是访问越界，于是产生越界中断信号。若未越界，则根据段表的始址和该段的段号，计算出该段对应段表项的位置，从中读出该段在内存的起始地址。
(2)然后，再检查段内地址d是否超过该段的段长SL。若超过，即d>SL，同样发出越界中断信号。若未越界，则将该段的基址d与段内地址相加，即可得到要访问的内存物理地址。图4-20示出了分段系统的地址变换过程。

3.4 页面置换算法

3.4.1 最佳置换算法

其所选择的被淘汰页面将是以后永不使用的，或许是在最长(未来)时间内不再被访问的页面。

无法实现

3.5 先入先出置换

算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰

3.6 最近最久未使用LRU

最近最久未使用(LRU)的页面置换算法是根据页面调入内存后的使用情况做出决策的。

LRU置换算法选择最近最久未使用的页面予以淘汰。该算法赋予每个页面一个访问字段，用来记录一个页面自上次被访问以来所经历的时间t。选取现有页面中t值最大的页面予以淘汰

3.7 最少使用LFU

在采用LFU算法时，应为在内存中的每个页面设置一个移位寄存器，用来记录该页面被访问的频率。该置换算法选择在最近时期使用最少的页面作为淘汰页。

3.8 Clock置换

当利用简单Clock算法时，只需为每页设置一位访问位，再将内存中的所有页面都通过链接指针链接成一个循环队列。

当某页被访问时，其访问位被置1。置换算法在选择一页淘汰时，只需检查页的访问位。如果是0，就选择该页换出；若为1，则重新将它置0，暂不换出，给予该页第二次驻留内存的机会，再按照FIFO算法检查下一个页面。将检查到队列中的最后一个页面时，若其访问位仍为1，则再返回到队首去检查第一个页面。

3.8.1 改进Clock

把同时满足这两个条件的页面作为首选淘汰的页面。由访问位A和修改位M可以组合成下面四种类型的页面:

1类（A=0，M=0）：表示该页最近既未被访问，又未被修改，是最佳淘汰页。
2类（A=0，M=1）：表示该页最近未被访问，但已被修改，并不是很好的淘汰页。
3类（A=1，M=0）：表示该页最近已被访问，但未被修改，该页有可能再被访问。
4类（A=1，M=1）：表示该页最近已被访问且被修改，该页可能再被访问。

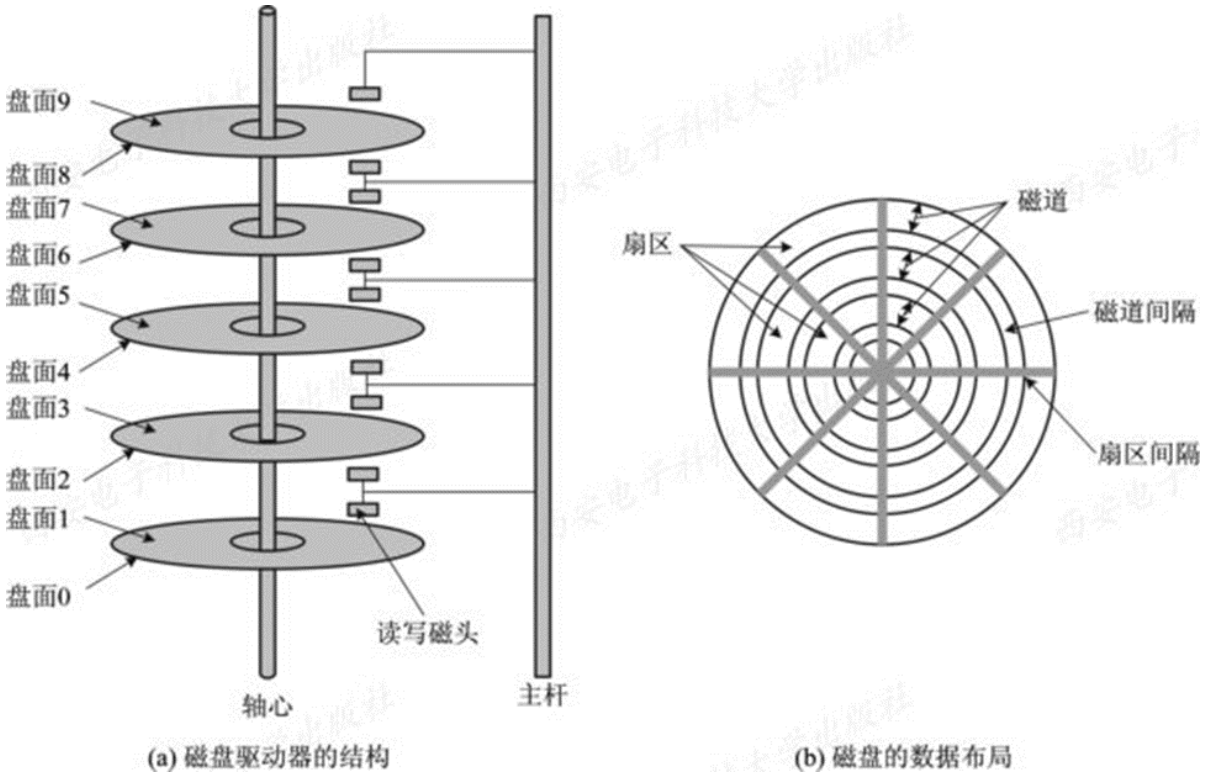
(1)从指针所指示的当前位置开始，扫描循环队列，寻找A=0且M=0的第一类页面，将所遇到的第一个页面作为所选中的淘汰页。**在第一次扫描期间不改变访问位A。**

(2)如果第一步失败，即查找一轮后未遇到第一类页面，则开始第二轮扫描，寻找A=0且M=1的第二类页面，将所遇到的第一个这类页面作为淘汰页。**在第二轮扫描期间，将所有扫描过的页面的访问位都置0。**

(3)如果第二步也失败，亦即未找到第二类页面，则将指针返回到开始的位置，并将所有的访问位复0。然后重复第一步，如果仍失败，必要时再重复第二步，此时就一定能找到被淘汰的页。

4. 磁盘调度

磁盘设备可包括一个或多个物理盘片，每个磁盘片分一个或两个存储面(Surface)(见图6-28(a))，每个盘面上有若干个磁道(Track)，磁道之间留有必要的间隙(Gap)。为使处理简单起见，在每条磁道上可存储相同数目的二进制位。



磁盘访问时间

Figure 7

磁盘设备在工作时以恒定速率旋转。为了读或写，磁头必须能移动到所指定的磁道上，并等待所指定的扇区的开始位置旋转到磁头下，然后再开始读或写数据。

磁盘的访问时间主要分为三个部分：

(1)寻道时间

(2)旋转延迟时间 **相对固定**

(3)传输时间 **相对固定**

4.1 先来先服务

4.2 最短寻道优先SSTF

该算法选择这样的进程，其要求访问的磁道与当前磁头所在的磁道距离最近，以使每次的寻道时间最短，但这种算法不能保证平均寻道时间最短。

4.3 基于扫描的磁盘调度算法

4.3.1 扫描算法SCAN

从内向外再从外向内，寻找同方向下最近的磁道

4.3.2 循环扫描算法CSCAN

磁头单向移动，如只从内到外，到尽头后再从内开始

4.4 NStepScan算法

为了解决进程反复提交io请求导致磁头一直停留在某些磁道上造成的**“磁臂黏着”**现象

按N个请求分为多个子队列

队列间按FCFS先来先服务，队列内按照SCAN算法执行

N趋于极大时，性能接近SCAN算法，N为1时，蜕化为FCFS算法

4.5 FSCAN算法

为了解决进程反复提交io请求导致磁头一直停留在某些磁道上造成的**“磁臂黏着”**现象

分为两个子队列，队列内按照SCAN算法执行

当前队列执行期间的新请求全部放入第二队列，两队列轮换