## 第八章

1、 CPU 暂停（Stall)：在读取内存数据时，CPU 空闲；
2、 地址绑定（重定位)：在程序载入内存时，把程序中的相对地址转换为内存中的绝对地址的过程
3、 指令和数据绑定到内存地址可在三个不同阶段：
编译时期（ Compile time）
如果内存位置已知，可生成绝对代码
如果开始位置改变，需要重新编译代码
加载时期（ Load time）
如果存储位置在编译时不知，则必须生成可重定位（ relocatable ）代码
执行时期（ Execution time）
如果进程执行时可在内存移动，则地址绑定可延迟到运行时
需要硬件对地址映射的支持（例如基址和限长寄存器）
4、 逻辑地址 Logical address
由 CPU 产生
在进程内的相对地址
也称：虚拟地址、程序地址
物理地址 Physical address
内存地址
所有内存统一编址
也称：绝对地址、实地址
5、 MMU：
把虚拟地址映射到物理地址的硬件；
是 CPU 用来管理内存的控制线路；
在 MMU 策略中,基址寄存器中的值在其送入内存的时候被加入到由一个用户进程所产生的每个地址中；
用户程序所对应到的是逻辑地址，物理地址对它从来都不可见。
6、 静态链接：在生成可执行文件时，把相关的二进制指令和数据包含在最终的可执行文件中。
动态链接：在程序运行时加载相关的二进制指令和数据。
7、存储管理的功能
1）内存分配
2）内存回收
3）地址绑定
4）存储保护
5）存储共享
8、分区(Hole)—可用的内存块，不同大小的分区分布在整个内存中。
9、连续内存管理——动态内存分配策略：
首次适应（ First-fit ）： 分配最先找到的合适的分区
最佳适应（Best-fit）： 搜索整个序列，找到适合条件的最小的分区进行分配
最差适应（ Worst-fit）： 搜索整个序列，寻找最大的分区进行分配
10、碎片：
外碎片 - 整个可用内存空间可以用来满足一个请求，但它不是连续的
内碎片 - 分配的内存可能比申请的内存大一点，这两者之间的差别是在分区内部，但又不被使用
11、紧缩：可通过紧缩来减少外碎片

把一些小的空闲内存结合成一个大的块。

只有重定位是动态的时候，才有可能进行紧缩，紧缩在执行时期进行

12、离散内存管理方案

1）分页 —现代操作系统常用方案

2）分段

3）段页式

13、分页（Paging）：

进程逻辑地址空间可能不连续。

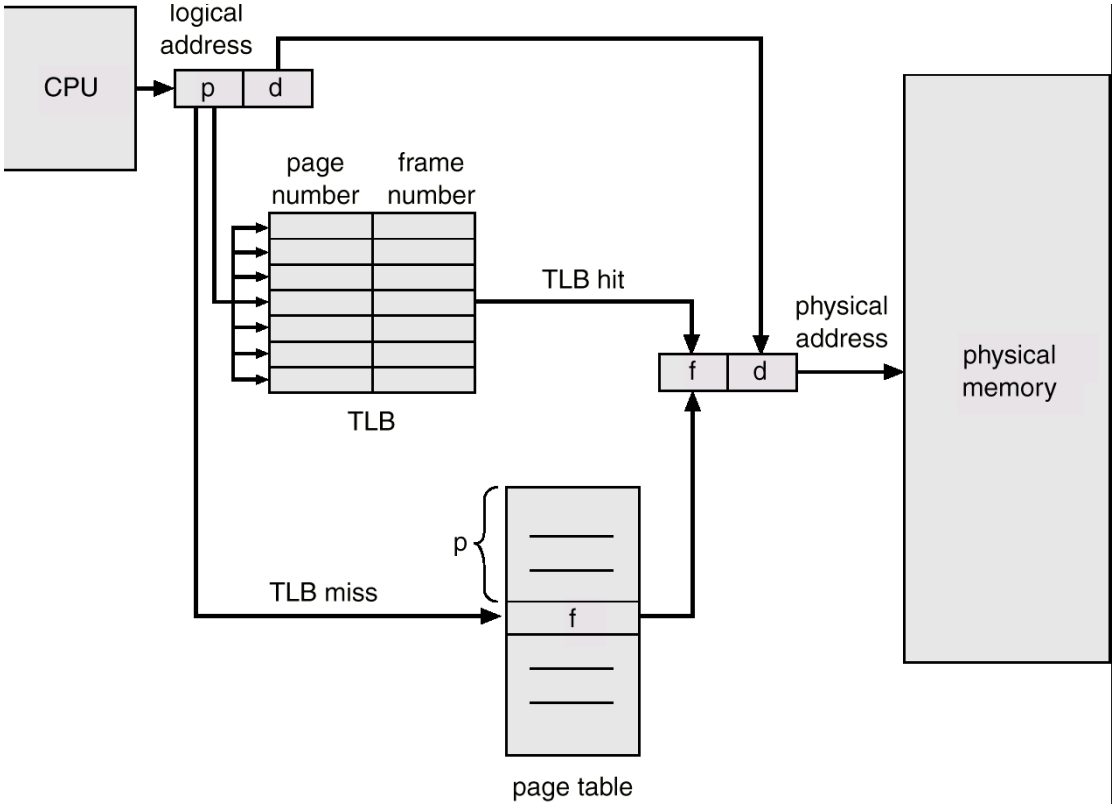
把物理内存分成大小固定的块，称为帧（Frame），大小为 2 的幂

把逻辑内存也分位固定大小的块，称为页（Page）

14、页表：

每一次的数据/指令存取需要两次内存存取，一次是存取页表，一次是存取数据/指令；

解决两次存取的问题，是采用小但专用且快速的硬件缓冲，这种缓冲称为转换表缓冲器(TLB)或联想寄存器。




15、有效访问时间 (EAT)：EAT = λ(a+b) + (1 - λ)(a+2b)

16、内存保护：

有效-无效位附在页表的每个表项中:

“有效”表示相关的页在进程的逻辑地址空间，并且是一个合法的页

“无效”表示页不在进程的逻辑地址空间中

17、页共享：

共享代码：

如果代码是可重入代码（只读），可以在进程间共享（如文本编辑器，编译器，window 系统）

共享代码必须出现在所有进程的逻辑地址空间的相同位置

18、页表

单级页表需要很多**连续**页面来存放页表不一定能实现。

解决方法：

1）层次页表

2）哈希页表

3）反向页表

19、分段：一个程序是一些段的集合，一个段是一个逻辑单位

一个逻辑地址是两个向量的集合:<segment-number, offset>

段表 - 映射二维物理地址，每个表项包括:

基址 - 包括内存中段物理地址的起始地址

限长 - 指定段的长度

20、段页式：先将用户程序分成若干个段，再把每个段分成若干个页，并为每个段赋予一个段名

三维地址：<段号，页号，页内偏移>

## 第九章

1、 局部性原理：一个程序只要部分装入内存就可以运行，整个程序不是同一时间都要运行。
2、程序部分装入技术优点：
1）进程大小不再受到物理内存大小限制
2）每个进程需要的内存更小
3）更多进程可以并发运行
4）I/O 更少
3、请求分页（按需调页）：只有在一个页需要的时候才把它换入内存。
缺页率：p
如果 p = 0 ，没有缺页
如果 p = 1， 每次访问都缺页
有效存取时间（EAT）：
EAT = (1 – p) x 内存访问时间 +
p(页错误时间 + [页交换出去时间] + 读入页时间 + 重启进程开销)
例题：
存取内存的时间= 200 nanoseconds (ns)
平均缺页处理时间 = 8 milliseconds (ms)
EAT = (1 – p) x 200 + p (8 milliseconds)
= (1 – p) x 200 + p x 8,000,000
4、写时复制（Copy-on-Write）允许父进程和子进程在初始化时共享页面
5、页置换算法：
FIFO： FIFO 算法可能会产生 Belady 异常。更多的页框->更多的缺页。
最优置换算法（Optimal Page Replacement, OPT/MIN）：
最近最少使用算法(LRU)：Counter 实现 或 Stack 实现；
LRU 近似算法：
1）附加引用位算法（左记录，右移位）；
2）二次机会算法：FIFO+引用位（1->0->置换）；
3）增强型二次算法：（引用位，修改位），分四类(0,0)(0,1)(1,0)(1,1)依次置换；
6、页框的分配：必须满足每个进程所需要最少的页数
平均分配：等大分配给每个进程；
比例分配：依据相对大小，相对优先级（优先级分配），或相对大小及优先级；
全局替换：进程在所有的页中选择一个替换页面；可以从另一个进程中获得页面。
局部替换：每个进程只从属于它自己的页中选择
7、颠簸：一个进程的页面经常换入换出
如果一个进程没有足够的页，那么缺页率将很高，这将导致：
CPU 利用率低下；
操作系统认为需要增加多道程序设计的道数；
系统中将加入一个新的进程。
8、内核内存分配：不同于用户内存
1）Buddy 系统：
从物理上**连续**的大小固定的段上分配内存
内存按**2 的幂**的大小来分配，即 4KB、8KB 等
优：通过合并快速生成更大的段；

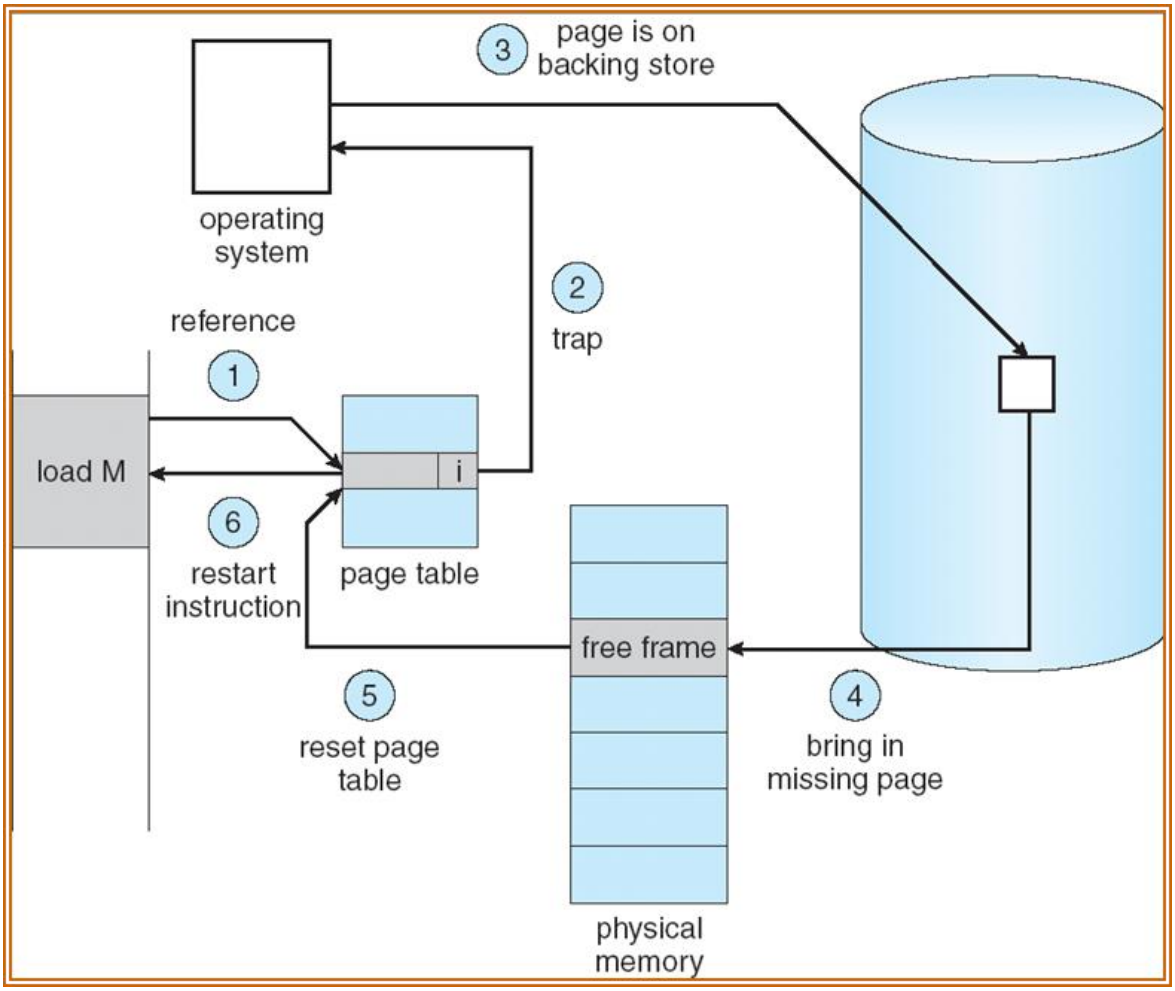
缺：按 2 的幂分配，易产生碎片；

2）Slab 分配：

Slab 是由一个或多个物理上**连续**的页组成。

每个内核数据结构都有一个 cache，Cache 含有一个或多个 slab。

每个内核数据结构所对应的 Cache 均预创建，使用时只需标记 free/full 即可。



## 第十章

1、目录：一个包含着所有文件信息的节点集合。
2、一个磁盘可有多个分区，一个分区可由多个磁盘组成。
3、目录性能：
如果
目录项大小= ds bytes
目录中最多文件数 = n
物理块大小 = b
那么
目录文件大小 = ds*n bytes
目录文件需要的物理块数目 = ds*n/b
搜索一个文件需要平均读入的块数=(ds*n/b+1)/2
因此
降低 ds -> 降低读块数
4、目录结构：
单层目录：所有用户的文件都在同一目录中；
缺点：命名的问题；分组的问题；
双层目录：每个用户有自己的目录结构；
优点：不同用户可以有相同文件名；高效的搜索；路径；
缺点：文件无法分组；
树形目录：
优点：高效的搜索；分组的可能；绝对或相对路径；
有向无环图目录：实现文件共享（符号链接 或 硬链接）
通用图目录：可以有环
5、如何保证无环？
1）仅允许指向文件的链接，而不允许指向子目录的链接
2）垃圾回收
3）每当加入一个新链接时，使用环路检测算法判断是否正确
6、文件系统安装：要访问一个文件系统，必须先安装它。
一个未安装的文件系统将被安装在一个安装点(mount point)上。
7、文件访问：
1）顺序访问
2）直接访问（相对访问）；
3）索引访问：兼容顺序文件和直接文件的优点；
8、权限：
1）所有者-读写执行；2）组用户-读写；3）公共用户-执行；

## 第十一章

1、磁盘特点：1）原地重写； 2）直接访问；

2、文件系统结构：

1）逻辑文件系统（Logical file system）： 管理文件系统中的元数据——

根据文件名管理文件目录

把文件名转换为文件 ID，文件句柄

管理 FCB 和目录

存储保护

优点：降低复杂性和冗余；

缺点：增加系统开销和降低性能；

例子：CD-ROM：ISO 9660；Unix：UFS, FFS；Windows：FAT, FAT32, NTFS；Linux（40 多种）： ext2，ext3

2）文件组织模块（File organization module）：管理文件、逻辑块和物理块；

逻辑块转换为物理块，翻译逻辑地址；

管理空闲空间和为文件分配块。

3）基本文件系统（Basic file system）：向设备驱动程序发送读写磁盘物理块指令；

4）I/O 控制，设备驱动程序 （Device Drivers）：位于 I/O 控制层，用于管理 I/O 设备

命令： “read drive1, cylinder 72, track 2, sector 10, into memory location 1060”

向硬件控制器发送专门的控制命令

## 第十二章

1、定位时间/随机访问时间（Positioning time，random-access time)：
寻道时间（seek time)：移动磁臂到所需柱面的时间；
旋转延迟(rotational latency)：等待扇区移动到磁臂下的时间

平均 I/O 时间 =访问延迟+传输时间 + 系统开销

例子：4KB 块， 7200 RPM 磁盘， 5ms 平均寻道时间， 1Gb/sec 传输率，0 .1ms 控制开销：
5ms + 1/(7200/60)sec + 4KB / 1Gb/sec + 0.1ms =
5.1ms + 4.17ms+4 / 131072 sec =
9.27ms + .03ms = 9.3ms

2、磁盘调度依据：将寻道时间减到最小；

并发是在同一时段发生，并行是在同一时刻发生