**存储器管理**

一．程序的装入和链接



1. 程序的装入：
2. 绝对装入方式(Absolute Loading Mode): 程序中所使用的绝对地址，既可在编译或汇编时给出， 也可由程序员直接赋予。
3. 可重定位装入方式(Relocation Loading Mode)



c.动态运行时装入方式(Denamle Run-time Loading):一边执行一边将逻辑地址转换为物理地址

1. 程序的链接
2. 静态链接方式(Static Linking):在程序运行之前，先讲各个目标模块链接成一个完整的装配模块，以后不在拆开



b.装入时动态链接(Loadtime Dynamic Linking):在装入内存时，采用边装入边链接。便于修改和更新。 便于实现对目标模块的共享。

c.运行时动态链接(Run-time Dynamic Linking):这种链接方式是将对某些模块的链接推迟到执行时才执行，亦即，在执行过程中，当发现一个被调用模块尚未装入内存时，立即由OS去找到该模块并将之装入内存， 把它链接到调用者模块上。凡在执行过程中未被用到的目标模块，都不会被调入内存和被链接到装入模块上，这样不仅可加快程序的装入过程，而且可节省大量的内存空间。

1. 连续分配方式
2. 单一连续分配：这是最简单的一种存储管理方式，但只能用于单用户、单任务的操作系统中。采用这种存储管理方式时，可把内存分为系统区和用户区两部分，系统区仅提供给OS使用，通常是放在内存的低址部分；用户区是指除系统区以外的全部内存空间， 提供给用户使用。
3. 固定分区分配:

(1)分区大小相等， 即使所有的内存分区大小相等。当程序太小时，浪费内存，当程序太小时，不能执行，适用于操作同一对象

(2)分区大小不等。

3.动态分区分配

(1)分区分配中的数据结构

空闲分区表：在系统设置一张空闲分区表，每条项目记录每个空闲分区的信息

空闲分区链：



(1)分区分配算法

a.首次适应算法FF

b.循环首次适应算法，该算法是由首次适应算法演变而成的。

c.最佳适应算法。

1. 可重定位分区分配
2. 动态重定位的引入



1. 动态重定位的实现

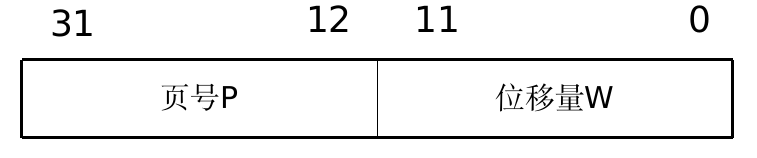


1. 对换(Swapping):中级调度

所谓“对换”， 是指把内存中暂时不能运行的进程或者暂时不用的程序和数据，调出到外存上，以便腾出足够的内存空间，再把已具备运行条件的进程或进程所需要的程序和数据，调入内存。对换是提高内存利用率的有效措施。

1. 基本分页存储管理方式
2. 页面:分页存储管理，是将一个进程的逻辑地址空间分成若干个大小相等的片，称为页面或页，并为各页加以编号，从0开始，如第0页、第1页等。由于进程的最后一页经常装不满一块而形成了不可利用的碎片，称之为“页内碎片”
3. 物理块：分页存储管理将内存空间分成与页面相同大小的若干个存储块，称为(物理)块或页框(frame)， 也同样为它们加以编号，如0＃块、1＃块等等。
4. 页面大小：在分页系统中的页面其大小应适中。页面若太小，一方面虽然可使内存碎片减小，从而减少了内存碎片的总空间， 有利于提高内存利用率，但另一方面也会使每个进程占用较多的页面，从而导致进程的页表过长，占用大量内存； 此外，还会降低页面换进换出的效率。
5. 地址结构

分页地址中的地址结构如下：



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[ A/L ]

D=[ A ] MOD L

1. 地址变换机构

a.基本的地址变换机构

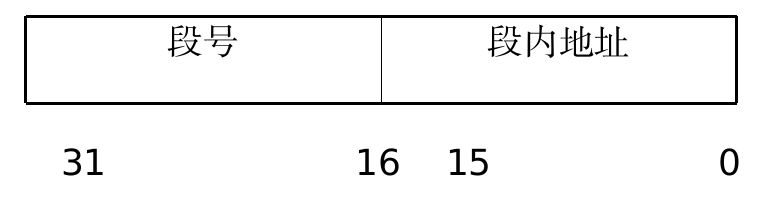


b.具有快表的地址变换机构



1. 基本分段存储管理方式
2. 分段系统的基本原理

在分段存储管理方式中，作业的[地址空间](http://baike.baidu.com/item/%E5%9C%B0%E5%9D%80%E7%A9%BA%E9%97%B4" \t "/home/wu/文档\\x/_blank)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](http://baike.baidu.com/item/%E5%AD%90%E7%A8%8B%E5%BA%8F" \t "/home/wu/文档\\x/_blank)段X、[数据段](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%AE%B5" \t "/home/wu/文档\\x/_blank)D及栈段S等，每个段都有自己的名字。为了实现简单起见，通常可用一个段号来代替段名，每个段都从0开始[编址](http://baike.baidu.com/item/%E7%BC%96%E5%9D%80" \t "/home/wu/文档\\x/_blank)，并采用一段连续的[地址空间](http://baike.baidu.com/item/%E5%9C%B0%E5%9D%80%E7%A9%BA%E9%97%B4" \t "/home/wu/文档\\x/_blank)。段的长度由相应的逻辑信息组的长度决定，因而各段长度不等。



1. 段表



1. 地址变换机构



1. 段页式存储管理方式
2. 基本原理



1. 地址变换过程



1. 虚拟存储器的基本概念

虚拟存储器， 是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所决定，其运行速度接近于内存速度，而每位的成本却又接近于外存。可见，虚拟存储技术是一种性能非常优越的存储器管理技术，故被广泛地应用于大、 中、 小型机器和微型机中。

1. 虚拟存储器的实现方法:
2. 分页请求系统：在分页系统基础上，增加了请求调页功能和置换功能，它只允许装入少量页数的程序，以后，在通过请求调页功能和置换功能将所需的页面装入

b.请求分段系统：在分段系统基础上，增加了请求调段功能和置换功能

1. 请求分页存储管理方式

1.请求分页中的硬件支持

(1)页表机制

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 页号 | 物理块号 | 状态位P | 访问字段A | 修改位M | 外存地址 |

状态位P:用于指示该页是否调入内存

访问字段A：用于记录本页在一段时间内被访问的次数，或记录该页有多长时间没有被访问

修改位M：表示该页调入内存后是否被修改过

(2)缺页中断机构

在请求分页系统中，每当所要访问的页面不在内存时，便产生缺页中断，请求OS将所缺页调入内存，他在指令执行期间产生和出理中断信号，一条指令执行期间，可能产生多个缺页中断

(3)地址变换机构



1. 内存分配
2. .最小物理块数的确定

是指能保证进程正常运行所需的最小物理块数。当系统为进程分配的物理块数少于此值时，进程将无法运行。

1. 物理块的分配策略
2. 固定分配局部置换(Fixed Allocation, Local Replacement) :

指为每个进程分配一定数目的物理块，在整个运行期间不会改变，当发现缺页时，从本进程的一页换出

1. 可变分配全局置换(Variable Allocation, Global Replacement) ：

指为每个进程分配一定数目的物理块，在整个运行期间，当发现缺页时，向系统的空闲链表中获取，唯有空闲链表没有空闲页，则向系统中的任意进程中换出一页

1. 可变分配局部置换(Variable Allocation, Local Replacemen):

指为每个进程分配一定数目的物理块，在整个运行期间，当发现缺页时，从本进程的一页换出，若发现置换频率过高，则os分配它一些空页，若发现置换频率过低，则os回收它一些空页

1. .物理块分配算法
2. 平均分配算法

这是将系统中所有可供分配的物理块，平均分配给各个进程。

1. 按比例分配算法

这是根据进程的大小按比例分配物理块的算法。

1. 考虑优先权的分配算法

在实际应用中，为了照顾到重要的、紧迫的作业能尽快地完成， 应为它分配较多的内存空间。通常采取的方法是把内存中可供分配的所有物理块分成两部分：一部分按比例地分配给各进程；另一部分则根据各进程的优先权，适当地增加其相应份额后，分配给各进程。

1. 调页策略
2. 何时调入页面

1)预调页策略:将预计在不久之后便会被访问的页面预先调入内存。本策略主要用于进程首次调入时，由程序员指定

2)请求调页策略:在程序运行期间，若发现有缺页，则发出请求中断，将所需页调入内存，该策略每次仅调入一页

(2)从何处调入页面

1) 系统拥有足够的对换区空间，这时可以全部从对换区调入所需页面，以提高调页速度

1. 系统缺少足够的对换区空间，这时凡是不会被修改的文件，都直接从文件区调入；但对于那些可能被修改的部分，在将它们换出时，便须调到对换区，以后需要时，再从对换区调入
2. UNIX方式。由于与进程有关的文件都放在文件区，故凡是未运行过的页面，都应从文件区调入。而对于曾经运行过但又被换出的页面，由于是被放在对换区，因此在下次调入时，应从对换区调入。由于UNIX系统允许页面共享，因此， 某进程所请求的页面有可能已被其它进程调入内存，此时也就无须再从对换区调入
3. 页面置换算法
4. 最佳(Optimal)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。采用最佳置换算法，通常可保证获得最低的缺页率。但不易预测

1. 先进先出(FIFO)页面置换算法

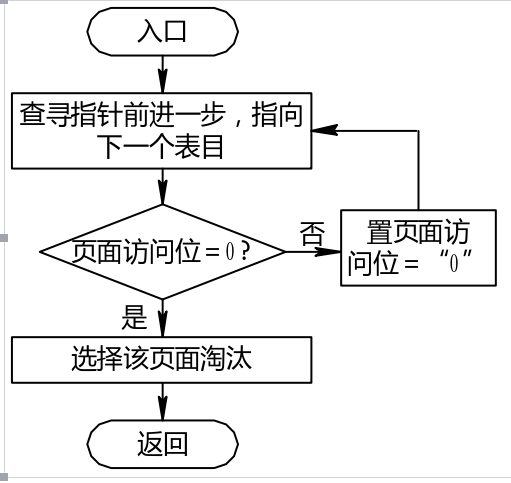
该算法总是淘汰最先进入内存的页面

(3)最近最久未使用(LRU)置换算法

由于无法预测各页面将来的使用情况，则选择最近最久没有使用的页面给予淘汰

1. Clock置换算法
2. 简单的Clock置换算法

只设置了一个访问位



b.改进型clock置换算法

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)： 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): 最近已被访问且被修改， 该页可能再被访问。

七．请求分段存储管理方式

1.请求分段中的硬件支持

(1)段表机制

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

1)存取方式：用于标识本分段的存取属性是只执行，只读，读写

2)存在位P:表示本段是否已调入内存

3)增补位：表示本段在运行过程是否做过动态增长

(2)缺段中断机构



1. 地址变换机构



2.分段的共享与保护

(1)共享段表



(2)共享段的分配

在为共享段分配内存时，对第一个请求使用该共享段的进程，由系统为该共享段分配一物理区，再把共享段调入该区，同时将该区的始址填入请求进程的段表的相应项中，还须在共享段表中增加一表项，填写有关数据，把count置为1；之后，当又有其它进程需要调用该共享段时，由于该共享段已被调入内存，故此时无须再为该段分配内存，而只需在调用进程的段表中，增加一表项，填写该共享段的物理地址；在共享段的段表中，填上调用进程的进程名、存取控制等，再执行count∶=count+1操作，以表明有两个进程共享该段

(3)共享段的回收

当共享此段的某进程不再需要该段时，应将该段释放， 包括撤在该进程段表中共享段所对应的表项，以及执行count∶=count-1操作。若结果为0，则须由系统回收该共享段的物理内存，以及取消在共享段表中该段所对应的表项， 表明此时已没有进程使用该段；否则(减1结果不为0)， 则只是取消调用者进程在共享段表中的有关记录。

(4)分段保护

1)越界检查 2) 存取控制检查

3) 环保护机构

一个程序可以访问驻留在相同环或较低特权环中的数据。

一个程序可以调用驻留在相同环或较高特权环中的服务