《操作系统》
实验指导书

南昌大学软件学院

实验 5：　段式存储管理模拟实验

5.1 实验概述

实验目的：

1. 了解段式存储的概念及实现原理；
2. 通过模拟实验深入体会段式存储段内地址到物理地址的转换关系。

实验语言：c

实验环境：linux、gcc

5.2 实验背景

一、段式虚存空间

段式存储管理把一个进程的虚拟地址空间设计成二维结构，即段号 s 与段内相对地址 w。与页式管理时不一样的是，页式管理中，被划分的页号按顺序编号递增排列，属一维空间，而段式管理中的段号与段号之间无顺序关系。另外，段的划分也不像页的划分那样具有相同的页长，段的长度的不固定的。每个段定义一组逻辑上完整的程序或数据。

每个段是一个首地址为零、连续的一维线性空间。根据需要，段长度动态增长。对段式虚地址空间的访问包括两个部分：段名和段内地址。其中的段名经编译程序和链接程序编译链接后转换成机器内部可以识别的段号和段内单元号。

二、段式存储管理的内存分配与释放

段式管理中以段为单位分配内存，每段分配一个连续的内存区。由于各段长度不等，所以这些存储区的大小不一。而且，同一进程所包含的各段之间不要求连续。

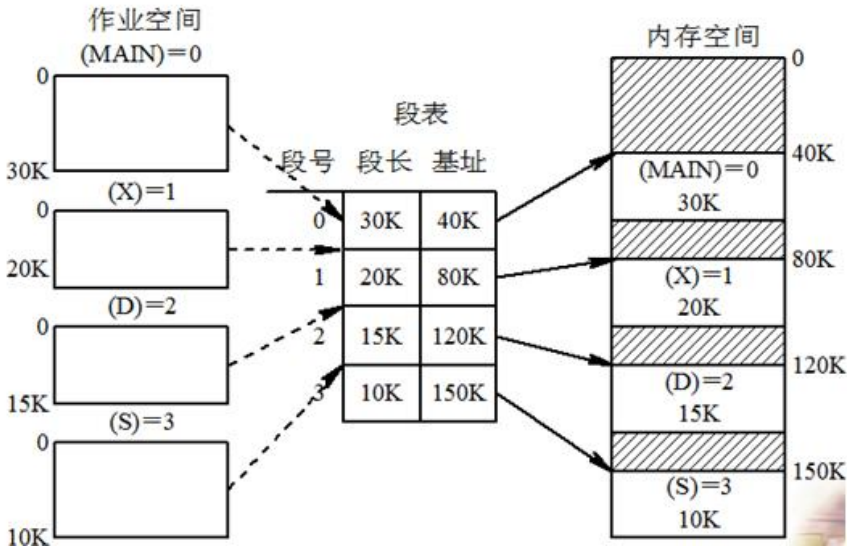
段式管理的内存分配与释放在作业或进程的执行过程中动态进行。动态分配过程是这样进行的，首先，段式管理为进程或作业分配部分内存，以作为该进程的工作区和放置即将执行的程序段。随着进程的执行，进程根据需要随时申请调入新段和释放老段。进程对内存区的申请和释放可分为两种情况：一种是当进程要求调入某一段时，内存中有足够的空闲区满足该段的内存要求；另一种是内存中没有足够的空闲区满足该段的内存要求。

除了初始分配之外，段的动态分配是在 CPU 所要访问的指令和数据不在内存时产生缺段中断的情况下发生的。

因此，段的淘汰或置换算法实际上是缺段中断处理过程的一部分。

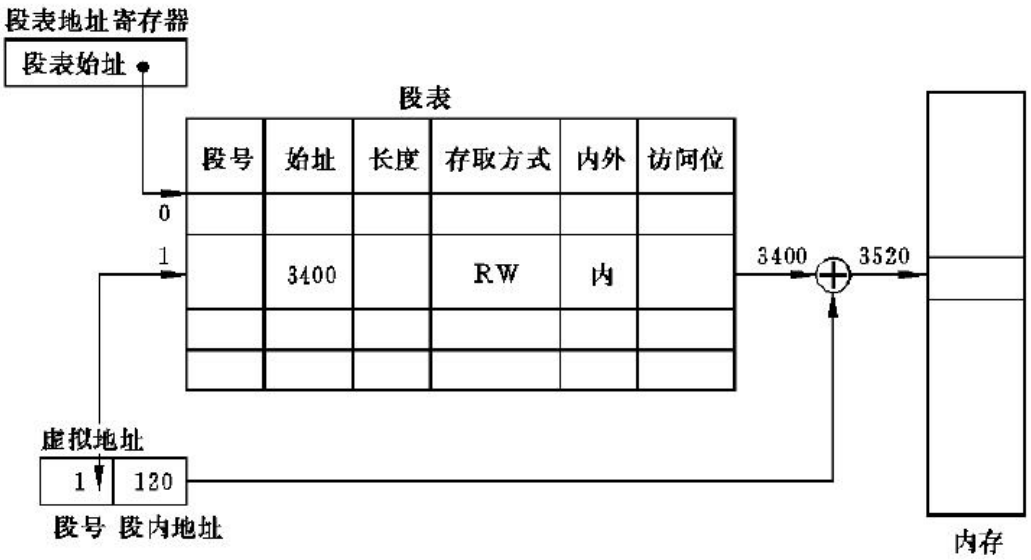
三、段式存储管理的地址变换

（1）段表



和页式管理方案类似，段式管理程序在进行初始内存分配之前，首先根据用户要求的内存大小为一个作业或进程建立一个段表。段式管理也是通过段表来进行内存管理。

（2）动态地址变换



一般在内存中给出一块固定的区域放置段表。当某进程开始执行时，管理程序首先把该进程的段表始址放入段表地址寄存器。通过访问段表寄存器，管理程序得到该进程的段表始址从而可开始访问段表。然后，由虚地址中的段号 s 为索引，查段表。若该段在内存，则判断其存取控制方式是否有错。如果存取控制方式正确，则从段表相应字段中查出该段在内存的起始地址，并将其和段内相对地址 w 相加，从而得到实际内存地址。如果该段不在内存，则产生缺段中断将 CPU

控制权交给内存分配程序。内存分配程序首先检查空闲区链，以找到足够长度的空闲区来装入所需要的段。如果内存中的可用空闲区总数小于所要求的段长时，则检查段表中访问位，以淘汰那些访问概率低的段并将需要段调入。

与页式管理时相同，段式管理时的地址变换过程也必须经过二次以上的内存访问。首先访问段表以计算得到待访问指令或数据的物理地址。然后才是对物理地址进行取数据或存数据操作。为了提高访问速度，页式地址变换时使用的高速关联存储器的方法也可以用在段式地址变换中。如果在关联存储器中找到了所需要的段，则可以大大加快地址变换速度。

四、段的共享与保护

（1）段的共享

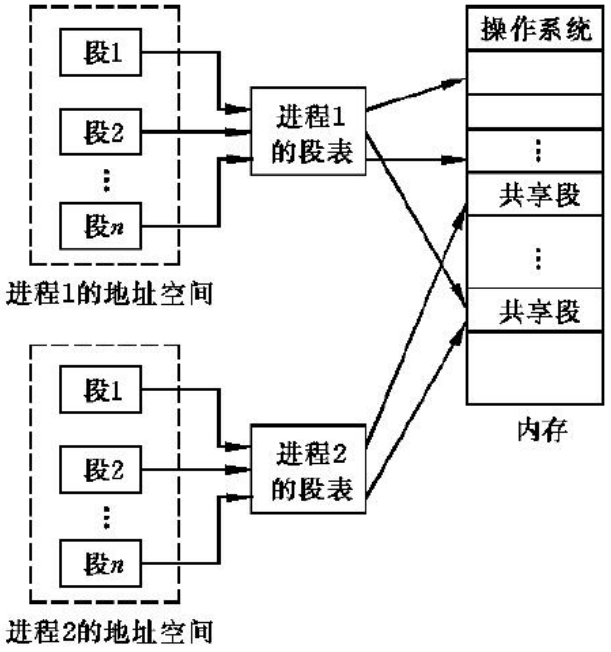
在多道环境下，常常有许多子程序和应用程序是被多个用户所使用的。如果每个用户进程或作业都在内存保留它们共享程序和数据的副本，那就会极大地浪费内存空间。最好的办法是内存中只保留一个副本，供多个用户使用，称为共享。一段程序为多个进程共享时：

①要求在执行过程中，该段程序的指令和数据不能被修改。

②在段表中设立相应的共享位来判别该段是否正被某个进程调用。显然一个正在被某个进程使用或即将被某个进程使用的共享段是不应该调出内存的。

（2）段的保护

与页式管理时相同，段式管理的保护主要有两种：地址越界保护法和存取方式控制保护法。



5.3 实验内容

一、相关数据结构

打开 segmmuhard.c，浏览代码可以发现定义了段(duan)和段表(duanbiao)两种数据结构。数据结构 duan 有三个成员，分别是该段长度，该段虚拟地址和该段物理地址，在该进程该段被调入内存后物理地址才起作用；数据结构 duanbiao 则是一个进程所占用的段及其他相关信息，第一个成员即该进程所占用的段，其它成员分别是进程的名字、标记进程是否被调入内存、该进程所占用的总段数和占用内存总量。

```
struct duan
{
    long capacity;  //该段的长度
    long addr;      //逻辑空间起始地址
    long realaddr;  //物理空间起始地址
};

struct duanbiao
{
    struct duan duans[10];
    char processname[20];   //进程的名字
    int isdiaoyong;         //是否被调用
    int num;                //段的数目
    long total;             //该进程的总占用量
};
```

二、相关函数实现

查看 main()函数可以发现定义了四种操作，分别是申请进程 apply()、显示进程 show()、调度进程 diaodu()和地址转换 zhuanhuan()。现在需要你根据函数的功能描述补全函数，然后编译运行代码验证程序的正确性。

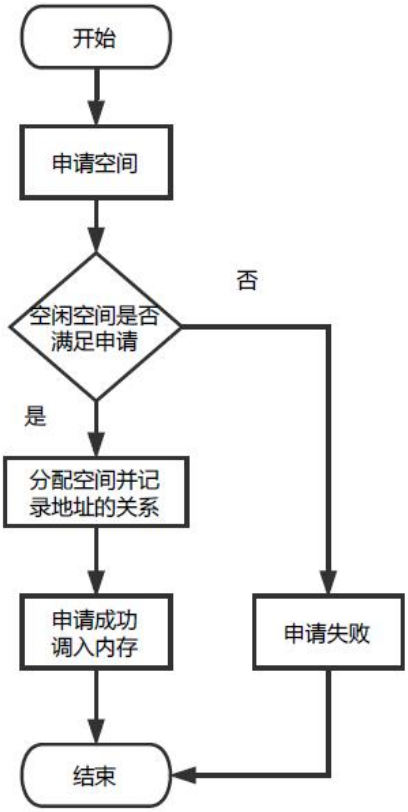
申请进程 apply()函数完成了新开进程的功能，同时还记录了该进程需要的内存空间段数和每段的具体大小，你需要补全该函数。

函数 diaodu()的主要功能是将某进程装入内存，在装入之前需要判断剩余可用空间是否能够满足内存需求，如果满足则将其装入内存，否则调入失败。在装入内存的过程中需要注意记录虚拟地址和物理地址的对应关系。在理解其功能的基础上，参考后面的流程图补全该函数。

函数 zhuanhuan()实则使用记录的虚拟地址与物理地址的关系，将你需要表示的某段段内偏移这一逻辑地址所对应的物理地址找到。现在也请你将缺失部分补全。

在补全程序之后，编译并运行代码验证程序的正确性，体会虚拟地址与物理地址的关系。

```
gcc segmmuhard.c -o segmmuhard
./segmmuhard
```



三、进一步

现在没有考虑淘汰旧的占用段，如果将其考虑进去，使用最简单的先进先出算法淘汰段该如何修改程序？