分页和分段的主要区别

a)、**页是信息的物理单位**，分页是为实现离散分配方式，以消减内存的外零头，提高内存的利用率；**段则是信息的逻辑单位，**它含有一组其意义相对完整的信息，分段的目的是为了能更好地满足用户的需要。

b)、**页的大小固定且由系统决定**，由系统把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而在系统中只能有一种大小的页面；**而段的长度却不固定，决定于用户所编写的程序**，通常由编译程序在对源程序进行编译时，根据信息的性质来划分。

c)、分页的作业地址空间是一维的，即单一的线性地址空间，程序员只需利用一个记忆符，即可表示一个地址；**而分段的作业地址空间则是二维的，程序员在标识一个地址是，即需给出段名，又需给出段内地址。**

引入**分段存储管理方式的目的**：满足**程序员在编程和使用上多方面的要求**。这种存储管理方式已经成为当今所有存储管理方式的基础。

1、分段存储管理方式的引入

主要满足用户和程序员以下需求：

1)、方便编程

用户把自己的作业按照逻辑管理划分为若干段，每个段都是从0开始编址，并有自己的名字和长度。因此，希望要访问的逻辑地址是由段名（段号）和段内偏移量（段内地址）决定的。

  LOAD1，[A] | <D>;//将分段A中D单元内的值读入寄存器1

  STORE1，[B] | <C>;//将寄存器1的内容存入B分段的C单元中

2)、信息共享

在实现对程序和数据的共享时，是以信息的逻辑单位为基础的。。比如共享某个例程和函数。分页系统中的页只是存放信息的物理单位（块），并无完整的意义，不便于实现共享；然而段却是信息的逻辑单位。由此可知，为了实现段的共享，希望存储管理能与用户程序分段的组织方式相适应。

3)、信息保护

4)、动态增长

有些段，会随着程序的使用不断增长。而事先又无法确切地知道数据段会增长到多大。

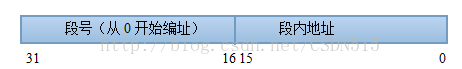
5)、动态链接

动态链接是指在作业运行前，并不把几个目标程序段链接起来。要运行时，先将主程序所对应的目标程序装入内存并启动运行，当运行过程中有需要调用某段时，才将该段调入内存并进行链接。可见动态链接也要求以段作为管理的单位。

2、分段系统的基本原理

2.1、分段

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。



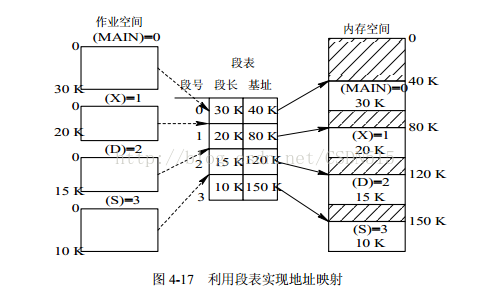
在该地址结构中，允许一个作业最长有64K个段，每个段的最大长度为64KB。

分段方式已得到许多编译程序的支持，编译程序能自动地根据源程序的情况而产生若各个段。

2.1、段表

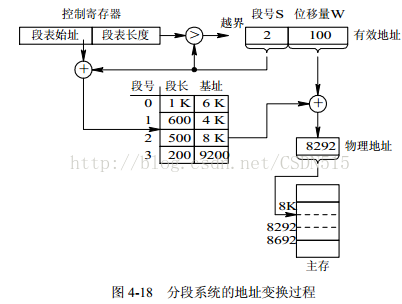
在动态分配方式中，系统为整个进程分配一个连续的内存空间。而在分段式存储管理系统中，则是为每个分段分配一个连续的分区，而进程中的各个段可以离散地移入内存不同的分区中。为了使程序能正常运行，也能从物理内存中找出每个逻辑段所对应的位置，应像分页那样，在系统中为每个进程建立一段映射表，简称“段表”。每个段在表中占有一个表项。其中记录了该段在内存中的起始地址（基址）和段的长度。段表可以存放在一组寄存器中，以提高访问速度，但更常见的是将段表放在内存中。

在配置了段表后，执行中的进程可通过查找段表找到每个段所对应的内存区。可见段表是用于实现从逻辑段到物理内存区的映射：



2.3、地址变换机构

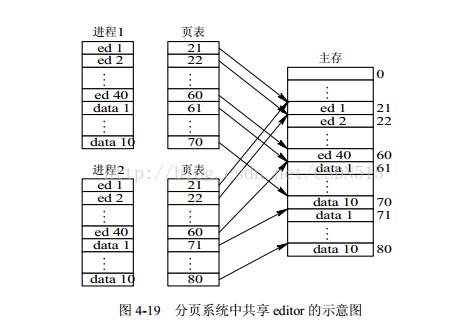
为了实现进程逻辑地址到物理地址的变换功能，在系统中设置了段表寄存器，用于存放段表起始地址和段表长度TL。在进行地址变换时，系统将逻辑地址中的段号S与段表长度TL进行比较。若S>TL，表示段号太大。访问越界，于是产生越界中断信号；若未越界，则根据段表的起始地址和该段的段号+段内地址从而到的要访问的内存物理地址。



与分页系统类似，当段表存放在内存中，每要访问一个数据，都需要访问两次内存，类似的，再增设一个联想寄存器，用于保存最近常用的段表项。

3、信息共享

在分段系统的允许若干个进程共享一个或多个分段，容易实现对段的保存和共享。



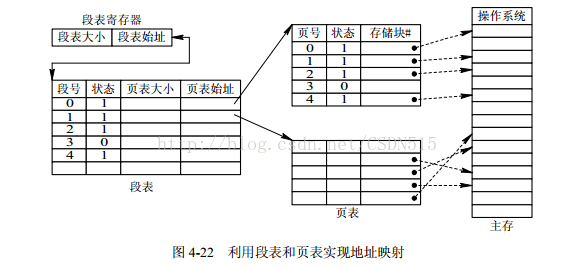
可重入代码又称为“纯代码”，是一种允许多个进程同时访问的代码。为使各个进程所执行的代码完全相同，绝对不允许可重入代码在执行中有任何的改变。在每个进程中，都必须配以局部数据区，把在执行中可能改变的部分拷贝到该数据区，这样程序在执行时，只需对数据区中的内容进行修改，并不去改变共享的代码，这时的可共享代码级成为可重入码。

4、段页式存储管理方式

**结合分页和分段的优点，分页系统很好解决内存的外碎片问题，分段系统有便于实现、分段可共享、易于保护、可动态链接等优点。结合这两种方式形成新系统称为“段页式系统”。**

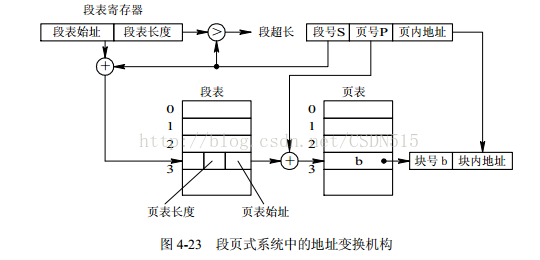
4.1、基本原理

**先分段，在段内进行分页，为每一个段赋予一个段名。以下展示出了一个作业地址空间的结构。该作业有三个段，页面大小4KB。**在段页式系统中，其地址结构由段号、段内页号及页内地址三部分所组成。

https://img-blog.csdn.net/20140503110945140?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvQ1NETjUxNQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

4.2、地址变换过程

为了方便段页式系统中地址变换的实现，需配置一个段表寄存器，其中存放段表起始地址和段表长TL。比较段号与TL是否越界，从段表寄存器中获取段表始址找到段表，根据段表内的页表始址找到对应的页表，在根据页表的存储块找到内存中的物理块，从而获取物理地址。



段页式系统中，为了获得一条指令或数据，须三次访问内存：

①　访问内存中的段表，从中取得页表始址

②　访问内存中的页表，从中取出该页所在的物理块号，并与页内地址形成物理地址

③　访问真正从第二次访问所得的地址中，取出指令或者数据

多次访问内存，执行速度降低，因此在地址变换机构中增设一个高速缓冲寄存器。每次访问它时，都须同时利用段号和页号去检索高速缓存，若找到匹配的表项，便可以从中得到相应页的物理块号，用来与页内地址一起形成物理地址；若未找到匹配表项，则仍需要再三次访问内存