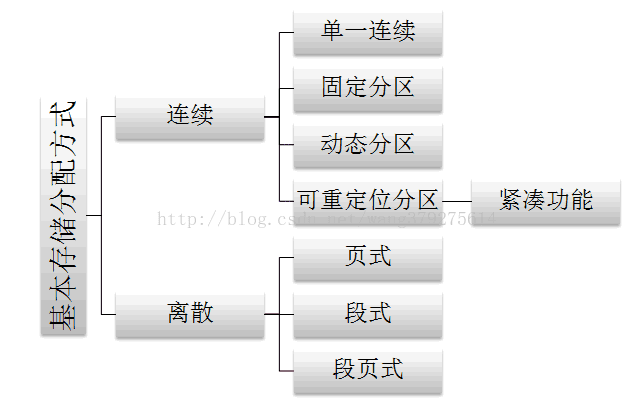
虚拟内存空间



1.  离散分配方式的出现

　　由于连续分配方式会形成许多内存碎片，虽可通过“紧凑”功能将碎片合并，但会付出很大开销。于是出现离散分配方式：将一个进程直接分散地装入到许多不相邻的内存分区中。

       下面主要介绍“离散分配”三种方式的基本原理以及步骤：

**2.  基本分页存储**

2.1.       步骤

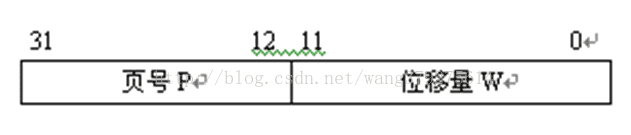
逻辑空间等分为页；并从0开始编号

内存空间等分为块，与页面大小相同；从0开始编号

分配内存时，以块为单位将虚拟内存中的若干个页分别装入到多个可以不相邻接的物理块中。

2.2.       逻辑地址结构

       分两部分：页号、位移量（页内地址）



      页内地址的位数可以决定页的大小（如上图每页大小为4K）。

　 逻辑地址=页号&位移量（&号是连接符号，是将页号作为逻辑地址的最高位）

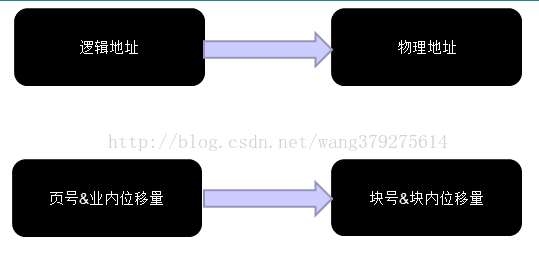
物理内存的地址结构：

块号 & 块内offset

物理地址中物理块的大小与逻辑地址中的页大小相等。

2.3.       地址映射（逻辑地址--->物理地址）

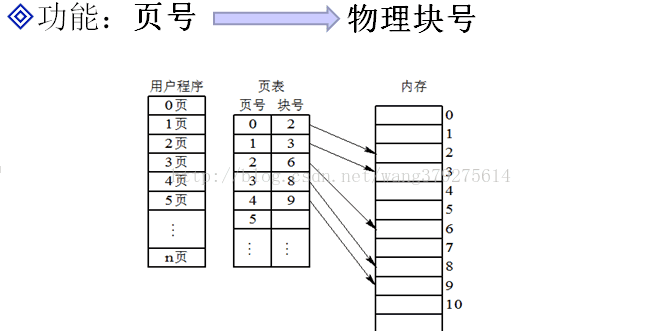
　　如下图所示：（物理地址=块号&块内地址）



　　因为块的大小=页的大小，所以块内位移量=页内位移量，所以只需求出块号即可：

如何求块号呢？页表来帮你

页表：（保存在内存中）



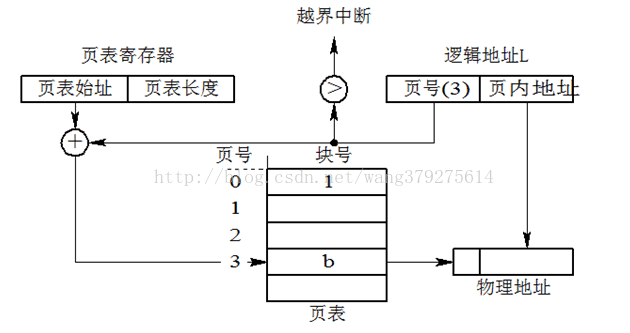
给定一个逻辑地址和页面大小，如何计算物理地址？

1)       根据页面大小可计算出页内地址的位数，从而将逻辑地址分割成页号和页内offset

2)       页号结合页表，即可得出块号

3)       块号&块内地址（等于页内地址）即可得出物理地址

2.4.       地址变换原理及步骤



请看上图，给出逻辑地址的页号和页内地址，开始进行地址变换：

1)       在被调进程的PCB中取出页表始址和页表大小（页表大小即虚拟内存的页号总长度），装入页表寄存器

2)       页号与页表寄存器的页表长度比较，若页号大于等于页表长度，发生地址越界中断，停止调用，否则继续

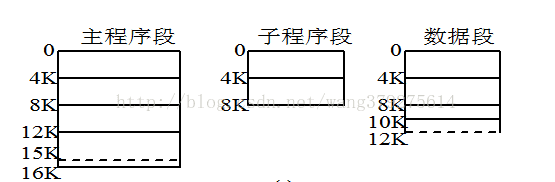
3)       由页号+页表始址求出块号

4)       块号&页内地址，即得物理地址

**3.   基本分段存储**

3.1.       步骤

³ 逻辑空间分为若干个段，每个段定义了一组有完整逻辑意义的信息（如主程序Main（）），如：



　　内存空间为每个段分配一个连续的分区

　　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，引入分段存储管理方式的目的主要是为了满足用户（程序员）在编程和使用上多方面的要求。

　　要注重理解，完整的逻辑意义信息，就是说将程序分页时，页的大小是固定的，只根据页面大小大小死生生的将程序切割开；而分段时比较灵活，只有一段程序有了完整的意义才将这一段切割开。（例如将一个人每隔50厘米切割一段，即为分页；而将一个人分割为头部、身体、腿部（有完整逻辑意义）三段，即为分段）

3.2.       地址结构

       分两部分：段号、位移量（段内地址）

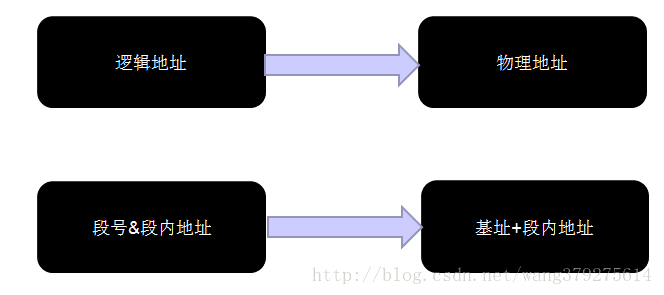


³  段内地址的位数可以决定段的大小

³  逻辑地址=段号&段内地址（&号是连接符号，是将段号作为逻辑地址的最高位）

3.3.       地址映射（逻辑地址--->物理地址）

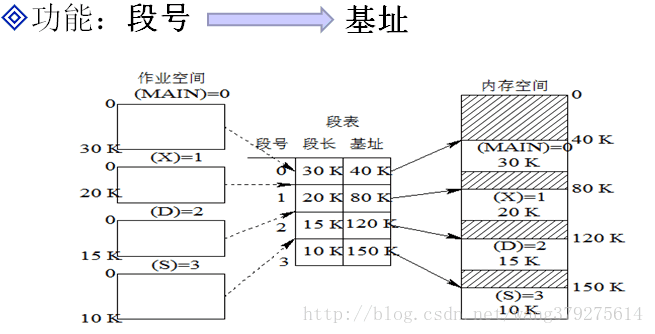
如下图所示：（物理地址=基址+段内地址）（注意为+号，而不是&号）



由上图可知若想求物理地址，只需求出基址即可：

如何求基址呢？段表来帮你

段表：（在内存里）



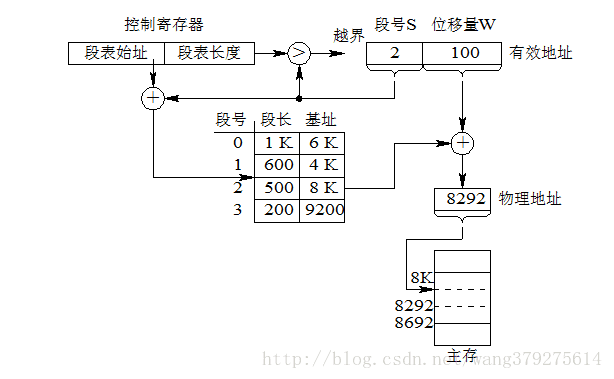
给定一个逻辑地址和页面大小，如何计算物理地址？

1)       根据逻辑地址，分割出段号和段内偏移

2)       由段号+段表基址得到基址

3) 基址+段内地址得到物理地址

3.4.       地址变换原理及步骤



请看上图，给出逻辑地址的段号和段内地址，开始进行地址变换：

1)       在被调进程的PCB中取出段表始址和段表长度，装入控制寄存器

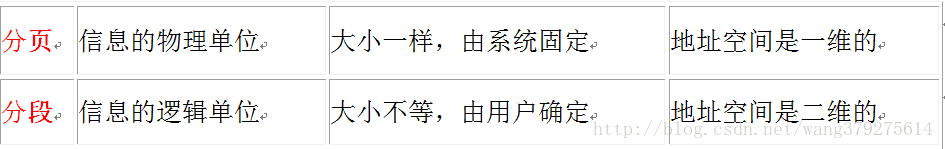
2)       段号与控制寄存器的段表长度比较，若段号大于等于段表长度，发生地址越界中断，停止调用，否则继续

3)       由段号+段表始址查询段表得到基址

4)       基址+段内地址，即得物理地址

以上即为段式存储的原理及整个过程……

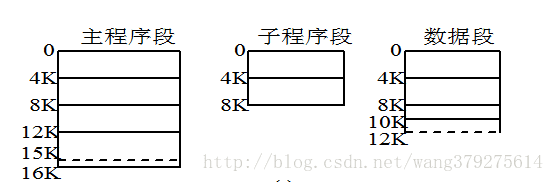
分页和分段的主要区别：



**4.  基本段页式存储**

4.1.       步骤

³ 用户程序先分段，每个段内部再分页（内部原理同基本的分页、分段相同）



4.2.       地址结构

       分三部分：段号、段内页号、页内地址



4.3.       地址映射（逻辑地址--->物理地址）

³ 逻辑地址----- >段号、段内页号、页内偏移

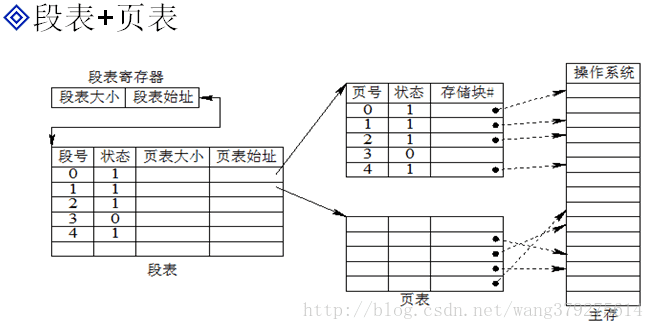
³ 段表寄存器--- >段表始址

³ 段号+段表始址---- 段表----->页表始址

³ 页表始址+段内页号----页表---->存储块号

³ 块号&页内地址------>物理地址

4.4.       地址变换原理及步骤



请看上图，给出逻辑地址的段号、段内页号、页内偏移(offset)，开始进行地址变换：

1)       在被调进程的PCB中取出段表始址和段表长度，装入段表寄存器

2)       段号与控制寄存器的段表长度比较，若段号大于等于段表长度，发生地址越界中断，停止调用，否则继续

3)       由段号结合段表始址求出页表始址和页表大小

4)       页号与段表的页表大小比较，若页号大于等于页表大小，发生地址越界中断，停止调用，否则继续

5)       由页表始址结合段内页号求出存储块号

6)       存储块号&页内偏移（块内偏移），即得物理地址

5.  总结

在页式、段式存储管理中，为获得一条指令或数据，须两次访问内存；

页式：

1. 访问内存，用页号查询页表，得到块号

返回，判断该块号的有效位

段式：

访问内存，用段号查询段表，得到基址

1. 访问内存，由物理地址得到data

而段页式则须三次访问内存

1. 访问内存，用段号查询段表，得到页表基址
2. 访问内存，用段内页号查询页表，得到块号
3. 访问内存，用物理地址得到data