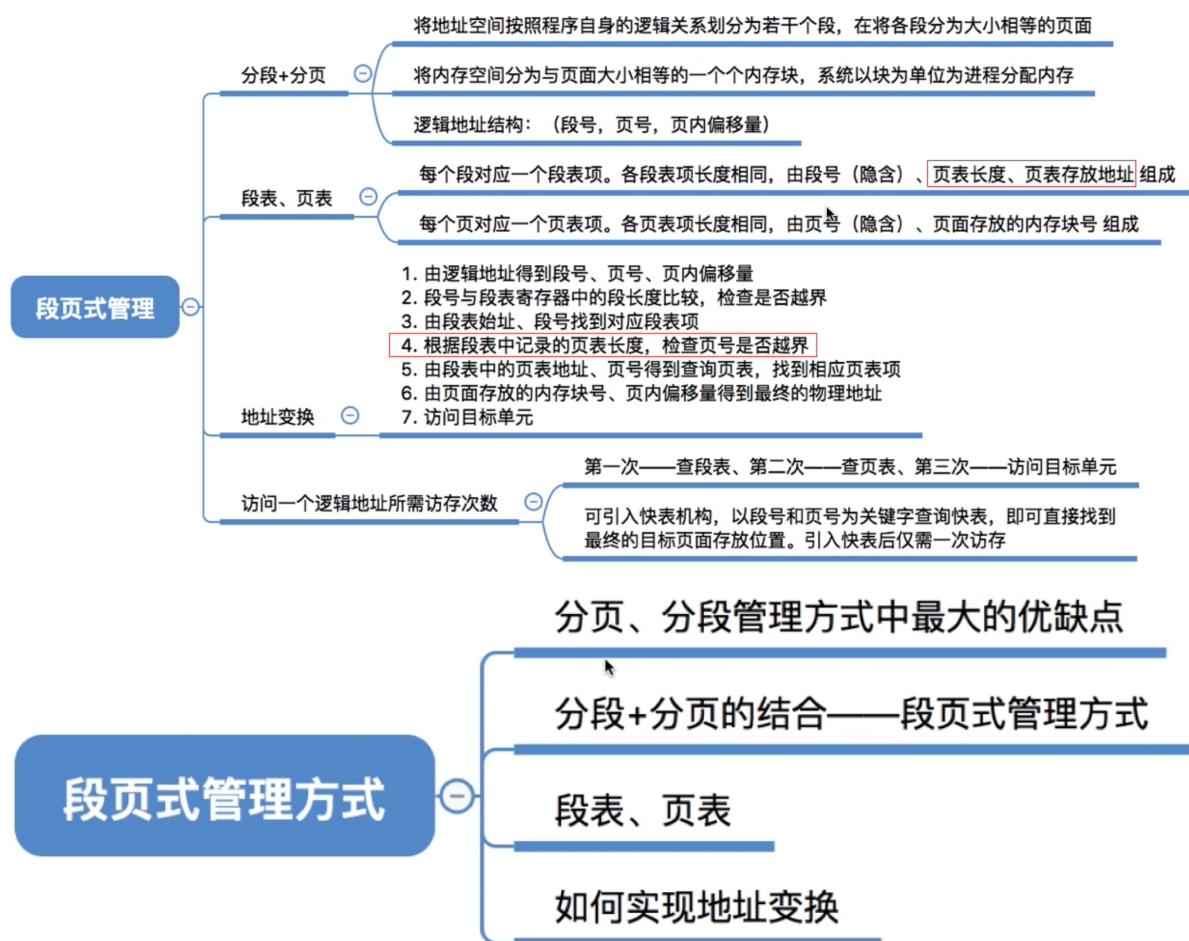


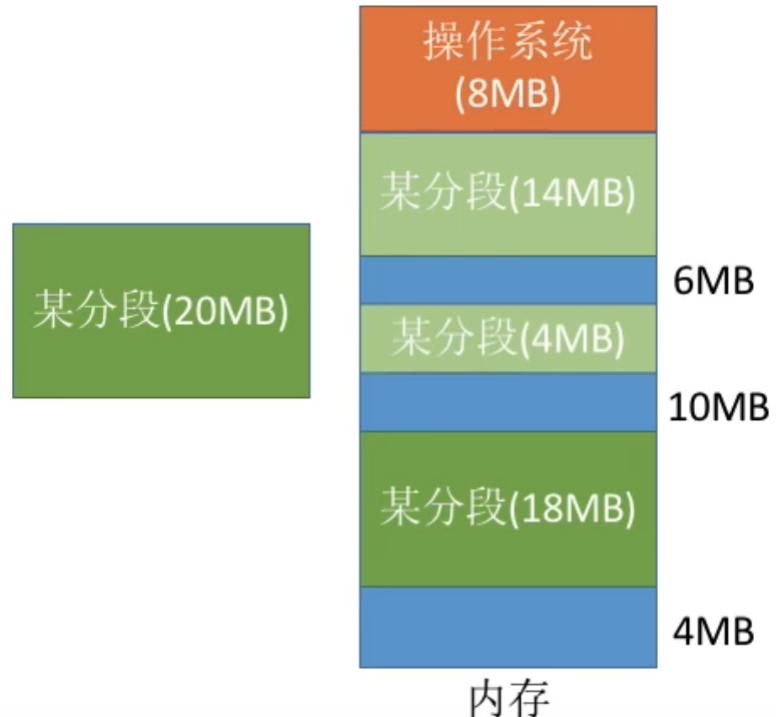
段页式管理方式



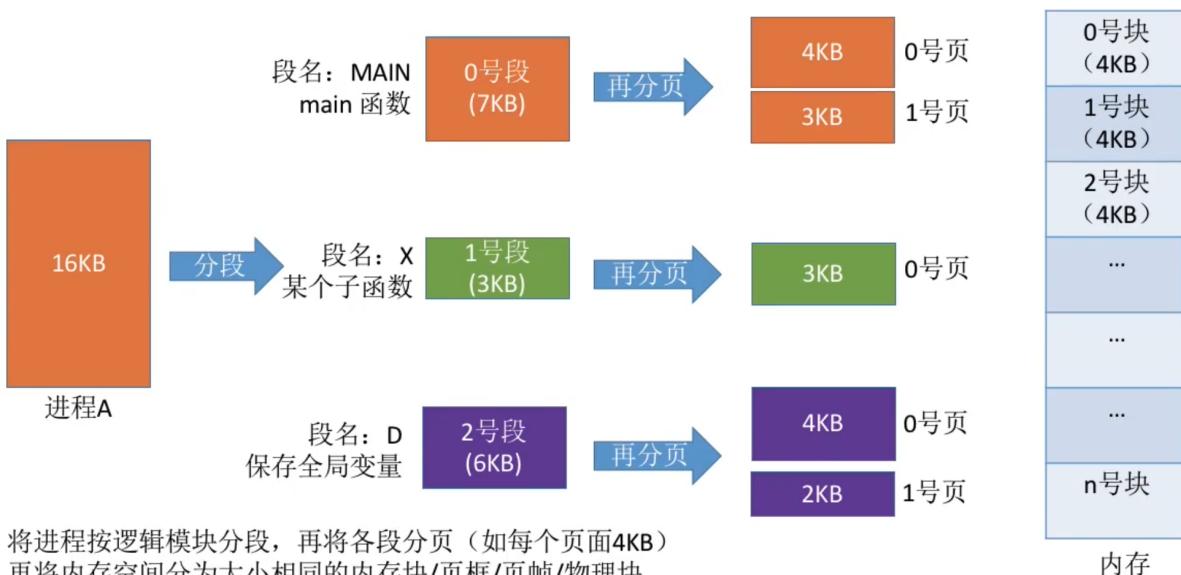
分页、分段的优缺点分析

	优点	缺点
分页管理	内存空间利用率高， 不会产生外部碎片 ，只会有少量的页内碎片	不方便按照逻辑模块实现信息的共享和保护
分段管理	很方便按照逻辑模块实现信息的共享和保护	如果段长过大，为其分配很大的连续空间会很不方便。另外，段式管理 会产生外部碎片

分段管理中产生的外部碎片也可以用“紧凑”来解决，只是需要付出较大的时间代价



分段+分页=段页式管理



段页式管理的逻辑地址结构

分段系统的逻辑地址结构由段号和段内地址（段内偏移量）组成。如：

31	16	15	0
段号	段内地址		

段页式系统的逻辑地址结构由段号、页号、页内地址（页内偏移量）组成。如：

31	16	15	12	11	0
段号	页号		页内偏移量		

段号的位数决定了每个进程最多可以分几个段

页号位数决定了每个段最大有多少页

页内偏移量决定了页面大小、内存块大小是多少

在上述例子中，若系统是按字节寻址的，则

段号占16位，因此在系统中，每个进程最多有

$$2^{16} = 64K \text{ 个段}$$

页号占4位，因此每个段最多有

$$2^4 = 16 \text{ 页}$$

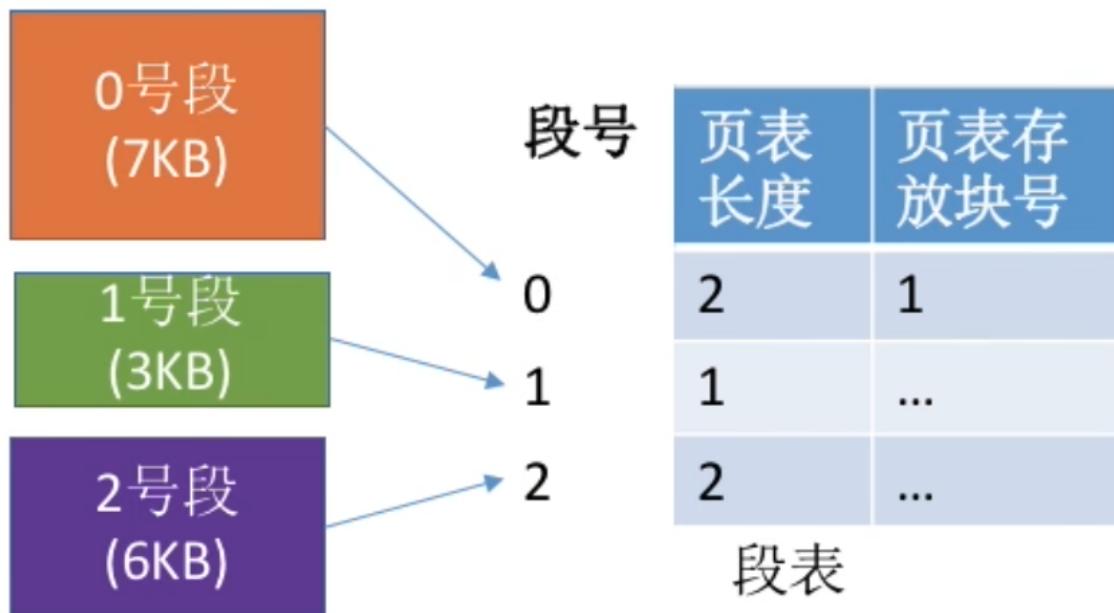
页内偏移量占12位，因此每个页面\每个内存块大小位

$$2^{12} = 4096 = 4KB$$

分段对用户是可见的，程序员编程时需要显式地给出段号、段内地址。而将各段分页对用户是不可见的。系统会根据段内地址自动划分页号和页内偏移量。

因此段页式管理的地址结构是二维的。

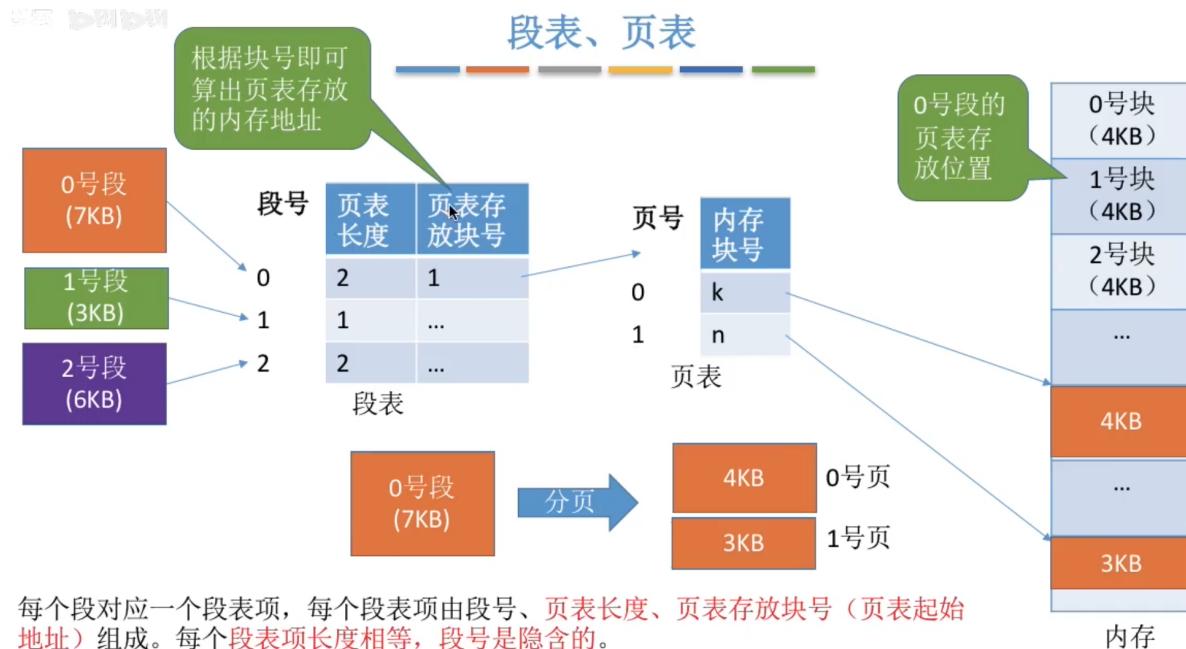
段表、页表



每个段对应一个段表项，每个段表项由段号、页表长度、页表存放块号（页表起始地址）组成。每个段表项长度相等，段号是隐含的。

根据块号即可算出页表存放的内存地址

每个页面对应一个页表项，每个页表项由页号、页面存放的内存块号组成。每个页表项长度相等，页号是隐含的。



每个段对应一个段表项，每个段表项由段号、**页表长度**、**页表存放块号**（页表起始地址）组成。每个**段表项长度相等**，**段号是隐含的**。

每个页面对应一个页表项，每个页表项由页号、页面存放的内存块号组成。每个页表项长度相等，页号是隐含的。

