**操作系统 请求段页式存储管理**

概述

* 请求段页式存储管理是建立在段页式存储管理基础上的一种段页式虚拟存储管理。
* 根据段页式存储管理的思想，请求段页式存储管理首先按照程序自身的逻辑结构，将其划分为若干个不同的分段，在每个段内则按页的大小划分为不同的页，内存空间则按照页的大小划分为若干个物理块。
* 内存以物理块为单位进行离散分配，不必将进程所有的页装入内存就可启动运行。
* 当进程运行过程中，访问到不在内存的页时，若该页所在的段在内存，则只产生缺页中断，将所缺的页调入内存；若该页所在的段不在内存，则先产生缺段中断再产生缺页中断，将所缺的页调入内存。若进程需要访问的页已在内存，则对页的管理与段页式存储管理相同。

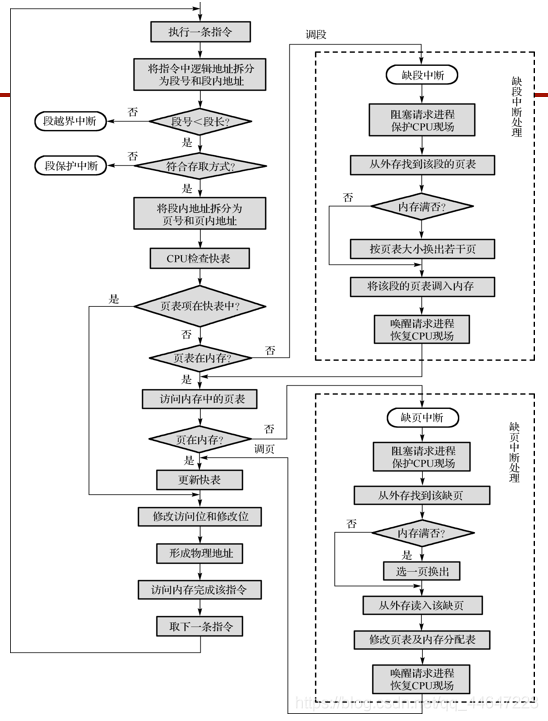
段表及页表机制

* 请求段页式存储管理中的页表和段表是两个重要的数据结构。
* 页表的结构与请求分页存储管理中的页表相似，段表则在段页式存储管理中的段表基础上增加了一些新的字段，这些新增的字段包括中断位（状态位）、修改位和外存始址等，用来支持实现虚拟存储器。

中断处理机制

* 内存空间的分配是以页为单位
  + 当某个进程在运行过程中发现所需页不在内存时，就先判断该页所在段的页表是否在内存，
  + 若页表已在内存，则只产生缺页中断将缺页由外存调入内存；
  + 若缺页所在段的页表不在内存则表明该段不在内存，这时先产生缺段中断由缺段中断处理程序为该段在内存中建立一张页表，将该页表的内存始址存入段表相应的段表项中，然后再产生缺页中断，将缺页由外存调入内存。
* 由于内存的分配是以页为单位进行的，产生中断一定是因为进程所访问的页不在内存，所以没有必要将包含该页的整个段调入内存，即在处理缺段中断时不必为整个段申请内存空间，于是在请求段页式存储管理中，缺段处理仅为所缺的段在内存中建立一张页表。
* 这种处理方式与请求分段存储管理中的缺段中断处理方式显然不同。

地址转换



————————————————

版权声明：本文为CSDN博主「爱吃鱼的喵996」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_44647223/article/details/111618870