6.2 按需调页

1. 基本概念

现在来考虑一个执行程序是如何从磁盘载入内存的。一种选择是在程序执行时，将整个程序载入到内存。不过，这种方法的间题是可能开始并不需要整个程序在内存中。如有的程序开始时带有一组用户可选的选工页。载入整个程序，也就将所有选硕的执行代码都载入到内存中，而不管这些选项是否使用。另一种选择是在需要时才调入相应的页。这种技术称为按需调页 (demand paging)，常为虚拟内存系统所采用。对于按需调页虚拟内存，只有程序执行需要时才载入页，那些从未访问的页不会调入到物理内存。

按需调页系统类似于使用交换的分页系统(见图 6.4)，进程驻留在第二级存储器上(通常为磁盘)。当需要执行进程时，将它换入内存。不过，不是将整个进程换入内存，而是使用懒惰交换(lazy swapper)。懒惰交换只有在需要页时，才将它调入内存。由于将进程看做是一系列的页，而不是一个大的连续空间，因此使用交换从技术上来讲并不准确。交换程序(swapper)对整个进程进行操作，而调页程序(pager)只是对进程的单个页进行操作。因此，在按需调页中，需要使用调页程序而不是交换程序。

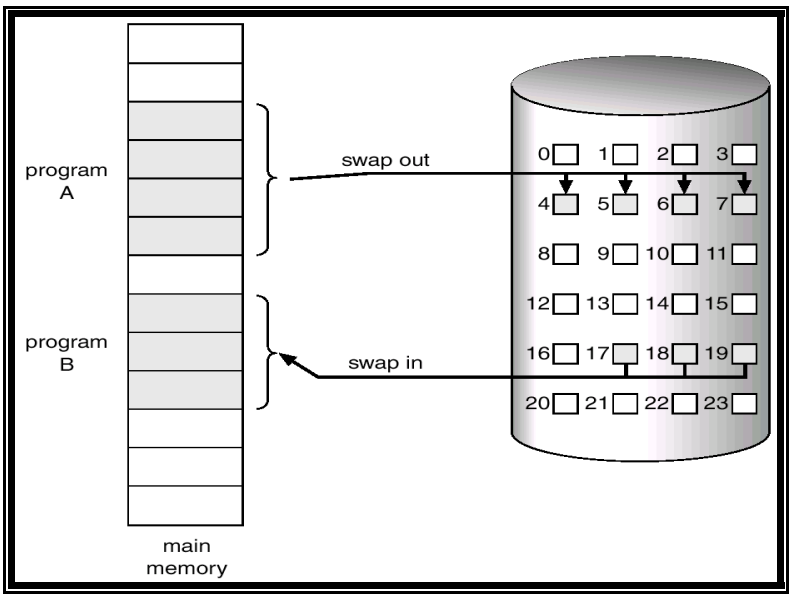

图 6.4 分页的内存到连续的磁盘空间之间的传送

当换入进程时，调页程序推测在该进程再次换出之前会用到哪些页。调页程序不是调入整个进程，而是把那些必需的页调入内存。这样，调页程序就避免了读入那些不使用的页，也减少了变换时间和所需的物理内存空间，减少了 I/O 和内存，从而更快地进行响应并且支持更多的用户。

2. 页表项的有效位

对按需调页这种方案，需要一定形式的硬件支持来区分哪些页在内存里，哪些页在磁盘上。有效-无效位(valid-invalid bit)可以用于这一目的。可以使用硬件在每一个页表的表项设置一个有效-无效位与该项相关联，并在初始化时都初始化为 0。当该位设置为 1 时，该值表示相关的页不仅合法且也在内存中。当该位设置为 0 时，该值表示相关的页为无效(也就是，不在进程的逻辑地址空间内)，或者有效但是在磁盘上。对于调入内存的页，其页表条目的

设置与平常一样；但是对于不在内存的页，其页表条目设置为无效，或包含该页在磁盘上的地址。这种情况如图 6.5 所示。

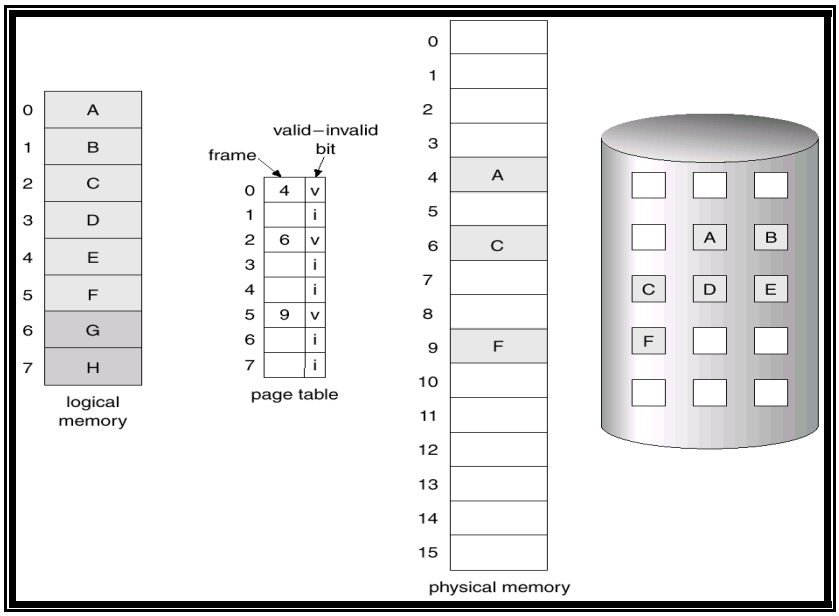

图 6.5 当有些页不在内存中时的页表

一种更完整的页表项硬件支持如下：

页号	物理块号	状态位P	访问字段A	修改位M	外存地址

图 6.6 一种更完整的页表项硬件支持

(1) 状态位 P(存在位)：用于指示该页是否已调入内存，供程序访问时参考。

(2) 访问字段 A：用于记录本页在一段时间内被访问的次数，或最近已有多长时间未被访问，提供给置换算法选择换出页时参考。

(3) 修改位 M：表示该页在调入内存后是否被修改过。由于内存中的每一页都在外存上保留一份副本，因此，若页未被修改，在置换该页时就不需将该页写回到外存上，这样可以减少系统的开销和启动磁盘的次数；若页已被修改，则必须将该页重写到外存上，以保证外存中所保留的始终是最新副本。

(4) 外存地址：用于指出该页在外存上的地址，通常是帧号，供调入该页时使用。

3. 缺页及处理流程

如果进程从不试图访问标记为无效的页，那么并没有什么影响。因此，如果推测正确并且只调入所有真正需要的页，那么进程就可如同所有页都已调入一样正常运行。当进程执行和访问那些驻留在内存中的页时，执行会正常进行。但是当进程试图访问那些尚未调入到内存的页时，标记为无效的访问会产生缺页陷阱(page-fault trap)。分页硬件在通过页表转换地址时，将发现无效位已设置，从而陷入操作系统。这种陷阱是由于操作系统未能将所需的页调入内存引起的。处理这种缺页的程序比较简单(见图 6.7)：

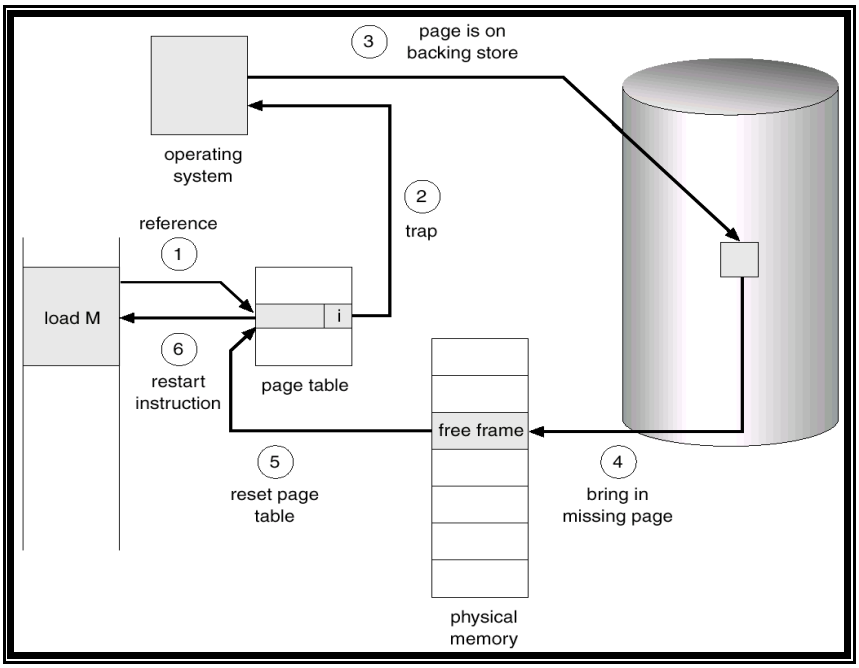

图 6.7 处理缺页的步骤

(1) 检查进程的内部页表(通常与 PCB 一起保存)，以确定该引用是合法还是非法的地址访问。

(2) 如果引用非法，那么终止进程。如果引用有效但是尚未调入页面，那么现在应调入。

(3) 找到一个空闲帧(例如，从空闲帧链表中选取一个)。

(4) 调度一个磁盘操作，以便将所需要的页调入刚分配的帧。

(5) 当磁盘读操作完成后，修改进程的内部表和页表(重新设置有效位为 1)，以表示该页已在内存中。

(6) 重新开始因陷阱而中断的指令。进程现在能访问所需的页，就好像它似乎总在内存中。

如果在处理缺页的时候找不到空闲帧该如何呢？这时候需要找到一些在内存中但当前没有被使用的页，将它们换出去。这就需要有一个算法来进行替换选择。通常需要一个能达成最小缺页数的算法，来换入/换出某些页面。在这种情况下，同一个页可能会被装入内存多次。

4. 按需调页的性能

按需调页对计算机系统的性能有重要影响。下面计算一下关于按需调页内存的有效访问时间(effective access time) 。对绝大多数计算机系统而言，内存访问时间(用 ma 表示)的范围为 10~200ns 。只要没有出现缺页，那么有效访问时间等于内存访问时间。然而，如果出现缺页，那么就必须先从磁盘中读入相关页，再访问所需要的字。

设 p 为缺页的概率($0 \le p \le 1.0$，p=0 代表没有缺页，p=1 代表每次都缺页)。希望 p 接近于 0，即缺页很少。那么有效访问时间为：

有效访问时间= (1-p)*ma + p*缺页时间

为了计算有效访问时间，必须知道处理缺页需要多少时间。缺页会引起如下序列的动作产生：

(1) 陷入到操作系统。

(2) 保存用户寄存器和进程状态。

(3) 确定中断是否为缺页。

(4) 检查页号|用是否合法并确定页所在磁盘的位置。

(5) 从磁盘读入页到空闲帧中。

a. 在该磁盘队列中等待，直到处理完读请求。

b. 等待磁盘的寻道和/或延迟时间。

c. 开始将磁盘的页传到空闲帧。

(6) 在等待时，将 CPU 分配给其他用户 (CPU 调度，可选)。

(7) 从 I/O 子系统接收到中断(以示 I/O 完成)。

(8) 保存其他用户的寄存器和进程状态(如果执行了第 6 步)。

(9) 确定中断是否来自磁盘。

(10) 修正页表和其他表以表示所需页现已在内存中。

(11) 等待 CPU 再次分配给本进程。

(12) 恢复用户寄存器、进程状态和新页表，再重新执行中断的指令。

以上步骤并不是在所有情况下都是必需的。例如，假设在第 6 步，在执行 I/O 时，将 CPU 分配给另一进程。这种安排允许多道程序以提高 CPU 使用，但是在执行完 I/O 时也需要额外的时间来重新启动缺页处理程序。

不管如何，缺页处理时间都包括以下三个主要部分：

(1) 处理缺页中断

(2) 读入页。

(3) 重新启动进程。

第 1 和第 3 个任务开销可以降低，如仔细编码，可只有数百条指令。这些任务每次可能只花费 1 ~ 100ms。另一方面，页切换时间可能接近 8ms (一个典型的磁盘的寻道时间为 5ms，延迟时间为 3ms，传输时间为 0.05 ms。因此，总的缺页处理时间可能为 8ms，包括硬件和软件时间)。而且，要注意这里只考虑了设备处理时间。如果有一队列的进程在等待设备(其他进程也引起了缺页)，那么必须加上等待设备的时间，这又增大了缺页处理时间。

设平均缺页处理时间为 8ms ，内存访问时间为 200ns ，那么有效内存访问时间(以 ns 计)为

有效访问时间 = (l - p) * 200ns + p * 8 ms

= (l - p) * 200ns + p * 8 000 000ns

= 200 + 7 999 800p (ns)

从上可以看出，有效访问时间与缺页率直接有关。如果每 1000 次访问中有一个缺页，那么有效访问时间为 8.2 间，即计算机会因为采用按需调页，而慢 40 倍。如果需要性能降低不超过 10% ，那么需要

220 > 200 + 7 999 800p

20 > 7 999 800p

p < 0.0000025

即为了让因缺页而出现的性能降低可以接受，只能允许每 399990 次访问中出现不到一次的缺页。总之，对于按需调页，降低缺页率是非常重要的。否则，有效访问时间会增加，会显著地降低进程的执行速度。同时，提高磁盘 I/O 的速度，对改善请求分页系统的性能至关重要。为此，应选用高速磁盘和高速磁盘接口作为请求分页的 I/O 设备。