3.6 有关实现的问题

实现虚拟内存系统要在主要的理论算法（如第二次机会算法与老化算法，局部页面分配与全局页面分配，请求调页与预先调页）之间进行选择。但同时也要注意一系列实际的实现问题。在这一节中将涉及一些通常情况下会遇到的问题以及一些解决方案。

3.6.1 与分页有关的工作

操作系统要在下面的四段时间里做与分页相关的工作：进程创建时，进程执行时，缺页中断时和进程终止时。下面将分别对这四个时期进行简短的分析。

当在分页系统中创建一个新进程时，操作系统要确定程序和数据在初始时有多大，并为它们创建一个页表。操作系统还要在内存中为页表分配空间并对其进行初始化。当进程被换出时，页表不需要驻留在内存中，但当进程运行时，它必须在内存中。另外，操作系统要在磁盘交换区中分配空间，以便在一个进程换出时在磁盘上有放置此进程的空间。操作系统还要用程序正文和数据对交换区进行初始化，这样当新进程发生缺页中断时，可以调入需要的页面。某些系统直接从磁盘上的可执行文件对程序正文进行分页，以节省磁盘空间和初始化时间。最后，操作系统必须把有关页表和磁盘交换区的信息存储在进程表中。

当调度一个进程执行时，必须为新进程重置MMU，刷新TLB，以清除以前的进程遗留的痕迹。新进程的页表必须成为当前页表，通常可以通过复制该页表或者把一个指向它的指针放进某个硬件寄存器来完成。有时，在进程初始化时可以把进程的部分或者全部页面装入内存中以减少缺页中断的发生，例如，PC（程序计数器）所指的页面肯定是需要的。

当缺页中断发生时，操作系统必须通过读硬件寄存器来确定是哪个虚拟地址造成了缺页中断。通过该信息，它要计算需要哪个页面，并在磁盘上对该页面进行定位。它必须找到合适的页框来存放新页面，必要时还要置换老的页面，然后把所需的页面读入页框。最后，还要备份程序计数器，使程序计数器指向引起缺页中断的指令，并重新执行该指令。

当进程退出的时候，操作系统必须释放进程的页表、页面和页面在硬盘上所占用的空间。如果某些页面是与其他进程共享的，当最后一个使用它们的进程终止的时候，才可以释放内存和磁盘上的页面。

3.6.2 缺页中断处理

我们终于可以讨论缺页中断发生的细节了。缺页中断发生时的事件顺序如下：

1) 硬件陷入内核，在堆栈中保存程序计数器。大多数机器将当前指令的各种状态信息保存在特殊的CPU寄存器中。

2) 启动一个汇编代码例程保存通用寄存器和其他易失的信息，以免被操作系统破坏。这个例程将操作系统作为一个函数来调用。

3) 当操作系统发现一个缺页中断时，尝试发现需要哪个虚拟页面。通常一个硬件寄存器包含了这一信息，如果没有的话，操作系统必须检索程序计数器，取出这条指令，用软件分析这条指令，看看它在缺页中断时正在做什么。

4) 一旦知道了发生缺页中断的虚拟地址，操作系统检查这个地址是否有效，并检查存取与保护是否一致。如果不一致，向进程发出一个信号或杀掉该进程。如果地址有效且没有保护错误发生，系统则检查是否有空闲页框。如果没有空闲页框，执行页面置换算法寻找一个页面来淘汰。

5) 如果选择的页框“脏”了，安排该页写回磁盘，并发生一次上下文切换，挂起产生缺页中断的进程，让其他进程运行直至磁盘传输结束。无论如何，该页框被标记为忙，以免因为其他原因而被其他进程占用。

6) 一旦页框“干净”后（无论是立刻还是在写回磁盘后），操作系统查找所需页面在磁盘上的地址，通过磁盘操作将其装入。该页面被装入后，产生缺页中断的进程仍然被挂起，并且如果有其他可运行的用户进程，则选择另一个用户进程运行。

7) 当磁盘中断发生时，表明该页已经被装入，页表已经更新可以反映它的位置，页框也被标记为正常状态。

8) 恢复发生缺页中断指令以前的状态，程序计数器重新指向这条指令。