第10章 内 存 管 理

本章介绍有关内存管理方面的内容，如虚拟内存的抽象模型和共享、按需调入页面、页面交换等。

10.1 内存管理的作用

内存管理系统是操作系统中最为重要的部分，因为系统的物理内存总是少于系统所需要的内存数量。虚拟内存就是为了克服这个矛盾而采用的策略。系统的虚拟内存通过在各个进程之间共享内存而使系统看起来有多于实际内存的内存容量。

虚拟内存可以提供以下的功能：

• 广阔的地址空间。

系统的虚拟内存可以比系统的实际内存大很多倍。

• 进程的保护。

系统中的每一个进程都有自己的虚拟地址空间。这些虚拟地址空间是完全分开的，这样一个进程的运行不会影响其他进程。并且，硬件上的虚拟内存机制是被保护的，内存不能被写入，这样可以防止迷失的应用程序覆盖代码的数据。

• 内存映射。

内存映射用来把文件映射到进程的地址空间。在内存映射中，文件的内容直接连接到进程的虚拟地址空间。

• 公平的物理内存分配。

内存管理系统允许系统中每一个运行的进程都可以公平地得到系统的物理内存。

• 共享虚拟内存。

虽然虚拟内存允许进程拥有自己单独的虚拟地址空间，但有时可能会希望进程共享内存。

10.2 虚拟内存的抽象模型

在讨论Linux系统虚拟内存的实现方法之前，让我们先看看虚拟内存的抽象模型。

当处理器执行一个程序时，它从内存中读取指令并解码执行。当执行这条指令时，处理器将还会需要在内存的某一个位置读取或存储数据。在一个虚拟内存系统中，所有程序涉及到的内存地址均为虚拟内存地址而不是机器的物理地址。处理器根据操作系统保存的一些信息将虚拟内存地址转换为物理地址。

为了让这种转换更为容易进行，虚拟内存和物理内存都分为大小固定的块，叫做页面。每一个页面有一个唯一的页面号，叫做PFN(page frame number)

在这种分页方式下，一个虚拟内存地址由两部分组成：一部分是位移地址，另一部分是PFN。每当处理器遇到一个虚拟内存地址时，它都将会分离出位移地址和 PFN地址。然后再将PFN地址翻译成物理地址，以便正确地读取其中的位移地址。处理器利用页面表来完成上述的工作。

图10-1是两个进程，进程 X和进程Y的虚拟内存示意图。两个进程分别有自己的页面表。

这些页面表用来将进程的虚拟内存页映射到物理内存页中。可以看出进程 X的虚拟内存页0映射到了物理内存页1，进程Y的虚拟内存页1映射到了物理内存4。页面表的每个入口一般都包括以下的内容：

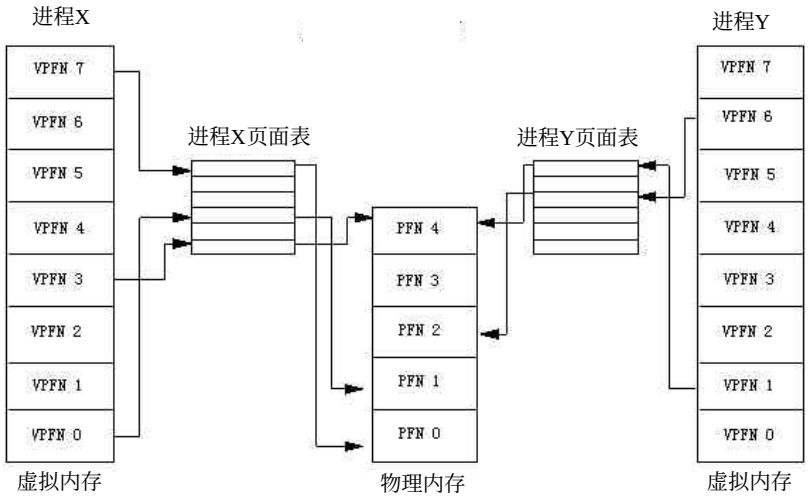

图10-1 虚拟内存示意图

• 有效标志。

此标志用于表明页面表入口是否可以使用。

• 物理页面号。

页面表入口描述的物理页面号

• 存取控制信息。

用来描述页面如何使用，例如，是否可写，是否包括可执行代码等。

处理器读取页面表时，使用虚拟内存页号作为页面表的位移，例如，虚拟内存页 5是页面表的第6个元素。

在将虚拟内存地址转换成物理内存地址时，处理器首先将虚拟内存地址分解为 PFN和位移值。例如，在上图中，一个页面的大小是0x2000 字节 (十进制的8192)，那么进程Y的一个虚拟内存地址0x2194将被分解成虚拟内存页号PFN为1和位移0x194。

然后处理器使用PFN作为进程页面表的位移值来查找页面表的入口。如果该入口是有效入口，处理器则从中取出物理内存的页面号。如果入口是无效入口，处理器则产生一个页面错误给操作系统，并将控制权交给操作系统。

假定此处是一个有效入口，则处理器取出物理页面号，并乘以物理页面的大小以便得到此物理页面在内存中的地址，最后加上位移值。

再看上面的例子：进程Y的PFN为1，映射到物理内存页号为4，则此页从0×8000 (4×0×2000)开始，再加上位移0×194，得到最终的物理地址为0×8194。

10.3 按需装入页面

由于物理内存要比虚拟内存小很多，所以操作系统一定要十分有效地利用系统的物理内存。一种节约物理内存的方法是只将执行程序时正在使用到的虚拟内存页面装入到系统的物理页面

中。当一个进程试图存取一个不在物理内存中的虚拟内存页面时，处理器将会产生一个页面错误给操作系统。如果发生页面错误的虚拟内存地址为无效的地址，说明处理器正在存取一个它不应该存取的地址。这时，有可能是应用程序出现了某一方面的错误，例如写入一个内存中的随机地址。在这种情况下，操作系统将会中止进程的运行，以防止系统中的其他进程受到破坏。

如果发生页面错误的虚拟内存地址为有效的地址，但此页面当前并不在物理内存中，则操作系统必须从硬盘中将正确的页面读入到系统内存。相对来说，读取硬盘要花费很长的时间，所以处理器必须等待直到页面读取完毕。如果此时有另外的进程等待运行，则操作系统将选择一个进程运行。从硬盘中读取的页面将被写入到一个空的物理内存页中，然后在进程的页面表中加入一个虚拟内存页面号入口。此时进程就可以重新运行了。

Linux系统使用需求时装入技术将可执行代码装入到进程的虚拟内存中。每当一个命令执行时，包括此命令的文件将被打开并映射到进程的虚拟内存中。此过程是通过修改描述进程内存映射的数据结构来实现的，通常被叫做内存映射。但此时只有文件镜像的第一部分被装入到了系统的物理内存中，而镜像的其他部分还保留在硬盘中。当此镜像执行时，处理器将产生页面错误，Linux使用进程的内存映射表决定应该把镜像的哪一部分装入到内存中执行。

10.4 交换

当一个进程需要把一个虚拟内存页面装入到物理内存而又没有空闲的物理内存时，操作系统必须将一个现在不用的页面从物理内存中扔掉以便为将要装入的虚拟内存页腾出空间。

如果将要扔掉的物理内存页一直没有被改写过，则操作系统将不保存此内存页，而只是简单地将它扔掉。如果再需要此内存页时，再从文件镜像中装入。

但是，如果此页面已经被修改过，操作系统就需要把页面的内容保存起来。这些页面称为“脏页面”（dirty page）。当它们从内存中移走时，将会被保存到一个特殊的交换文件中。

Linux系统使用一种叫做“最近最少使用”的技术（LRU）来决定把哪一个页面从物理内存中移出。系统中的每一个页面都有一个年龄，当一个页面被存取时，它的年龄将发生变化。页面被存取的越频繁，页面的年龄就越年轻；页面被存取的越少，它的年龄就越大。年龄大的页面将首先被交换出去。

10.5 共享虚拟内存

由于使用了虚拟内存，则几个进程之间的内存共享变得很容易。每个内存的存取都要通过页面表，而且每个内存都有自己的单独的页面表。如果希望两个进程共享一个物理内存页，只需将它们页面表入口中的物理内存号设置为相同的物理页面号即可。

10.6 存取控制

页面表中还包括存取控制信息，这样，在处理器使用页面表把进程的虚拟内存地址转换为物理内存地址时，可以方便地使用存取控制信息来检查进程是否存取了它不该存取的信息。

使用存取控制信息是完全必要的。例如，一些内存中包括可执行代码，而这些可执行代码通常为只读，操作系统将不会允许向一段只读代码中写入数据。同样，包括数据的内存通常为可读写的，而试图执行此内存中的代码将产生错误。大多数的处理器有内核和用户两种可执行方式。用户不能执行内核的代码，而内核的数据用户也无法存取。

10.7 高速缓存

为了获得最大的系统效用，操作系统一般使用高速缓存来提高系统性能。 Linux系统使用了几种涉及到高速缓存的内存管理方法。

10.7.1 缓冲区高速缓存

缓冲区高速缓存中保存着块设备驱动程序所用到的数据缓冲区。

这些缓冲区的大小固定，一般包括从块设备中读入的和将要写入到块设备中的信息块。快设备一次只能处理大小固定的数据块。硬盘就是块设备中的一种。

缓冲区高速缓存使用设备标识符和块号作为索引来快速地查找数据块。块设备只通过缓冲区高速缓存进行存取。如果所需要的数据存在于缓冲区高速缓存中，那么就不需要从物理块设备中读取，这样存取的速度就会加快。

10.7.2 页面高速缓存

页面高速缓存用来加速磁盘中文件镜像和数据的存取，我们将在后面的 10.12节中对它进行详细讨论。

10.7.3 交换高速缓存

交换文件中只保存那些被修改过的页面。

只要在页面被写入到交换文件中后没有被修改过，那么此页面下一次从内存中交换出来时就不用再写入到交换文件中了，因为交换文件中已经有了该页面。这样，该页面就可以简单地扔掉，节省了大量的系统操作。

10.7.4 硬件高速缓存

一个常用的硬件高速缓存是在处理器中，它一般保存着页面表的入口。

10.8 系统页面表

Linux系统共有三级的页面表。上一级页面表的PFN指向下一级页面表的入口。如图10-2所示，一个虚拟内存地址分成了几个字段，每个字段提供一个相应的页面表位移值。要把一个虚

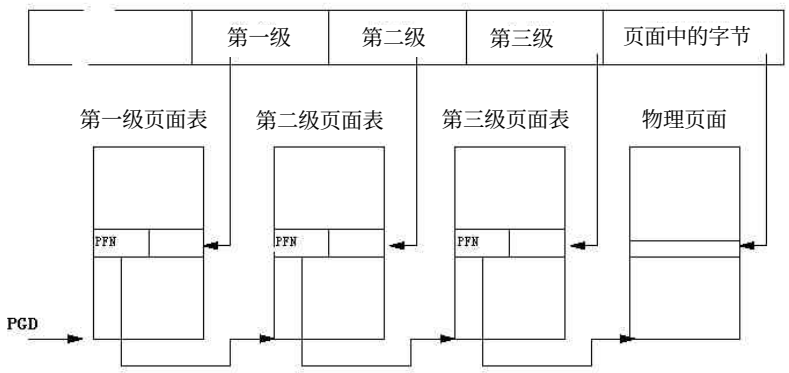

图10-2 虚拟内存地址示意图

拟内存地址翻译成一个物理内存地址，处理器必须读取每一个字段的内容，把其转化成包含页面表的物理页的位移值，再取出其中指向下一个页面表的 PFN。这个过程重复三次，直到找到包含物理页面号的虚拟内存地址。

Linux运行的所有平台都提供地址翻译的宏函数以便内核为某一个进程转化页面表。这样，内核就无需知道页面表入口的格式以及它们是如何安排的。

10.9 页面的分配和释放

系统在运行时会经常地需要物理内存页。例如，当一个文件镜像从磁盘调入到内存时，操作系统需要为它分配物理内存页。当程序执行完毕时，操作系统需要释放内存页。物理页的另一个用途是存储内核所需要的数据结构，例如页面表。页面的分配和撤消机制以及所涉及的数据结构对内存管理来说是至关重要的。

系统中所有的物理内存页都包括在 mem_map数据结构中，而mem_map是由mem_map_t结构组成的链表。mem_map_t在系统启动时初始化。每个mem_map_t结构都描述了系统中的一个物理页。其重要的字段有：

• count

此字段用于记录使用此页面的用户数。当几个进程共享此物理内存页时， count的值将会大于1。

• age

此字段描述了页面的年龄，通过此字段，操作系统可以决定是否将此页面扔掉或交换出去。

• map_nr

此字段为页面的物理页面号。

页面分配程序使用free_area向量查找和释放页面。对于页面分配程序来说，页面本身的大

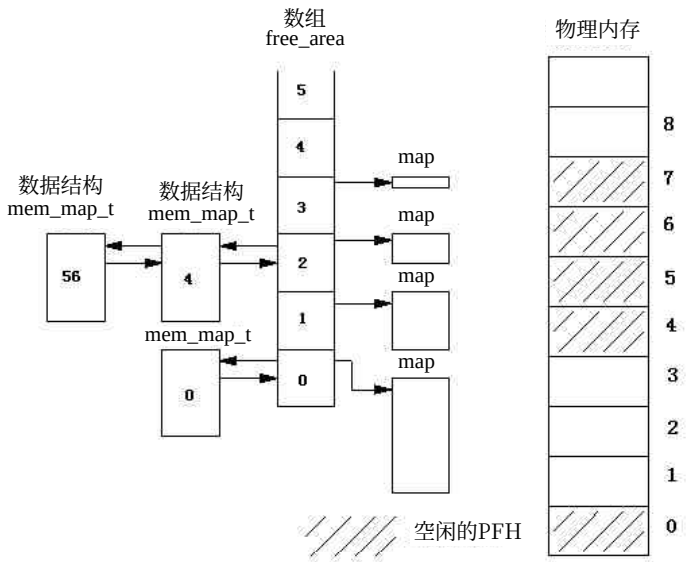

图10-3 空闲内存示意图

小和处理器使用的物理页面机制是无关的。

free_area中的每个元素都包括页面块的信息。第一个元素描述了单个页面的页面块，第二个元素描述了两个页面的页面块，第三个元素描述了四个页面的页面块，以次类推以 2的次方数增加。向量中的list元素用来作为指向mem_map数据结构中page结构的队列的头指针，指向空闲的页面。指针 map指向同样大小页面组的一个位图。如果第 N个页面块是空闲的，那么该位图的第N位置1。

图10-3显示了free_area结构。元素0有一个空闲页块（页号0），元素2有两个空闲页块，一个从页号4开始，另一个从页号56开始。

10.9.1 页面的分配

Linux系统使用Buddy算法来分配和释放页面块。如果系统对于请求的分配有足够的空闲页面(nr_free_pages > min_free_pages)，页面分配程序将会查找free_area以便找到一个和请求的页面块大小相同的页面块。它根据 free_area中list元素指向的空闲页面队列进行查找。如果没有同样大小的空闲页面块，则继续查找下一个空闲页面块（其大小为上一个页面块的 2倍）。如果有空闲的页面块，则把页面块分割成所请求的大小，返回到调用者。剩下的空闲页面块则插入在空闲页面块队列中。

10.9.2 页面的释放

以上页面块的分配策略会造成将一个个大的内存块分割成小块的结果。而内存页面释放程序却总是试图将一个个的比较小的页面块合并为大的页面块。

每当一个页面块释放时，页面释放程序就会检查其周围的页面块是否空闲。如果存在空闲的页面块，则空闲的页面块就会和释放的页面块合并在一起组成更大的页面块。

10.10 内存映射

当执行一个文件镜像时，可执行镜像的内容必须装入到进程的虚拟地址空间。可执行镜像

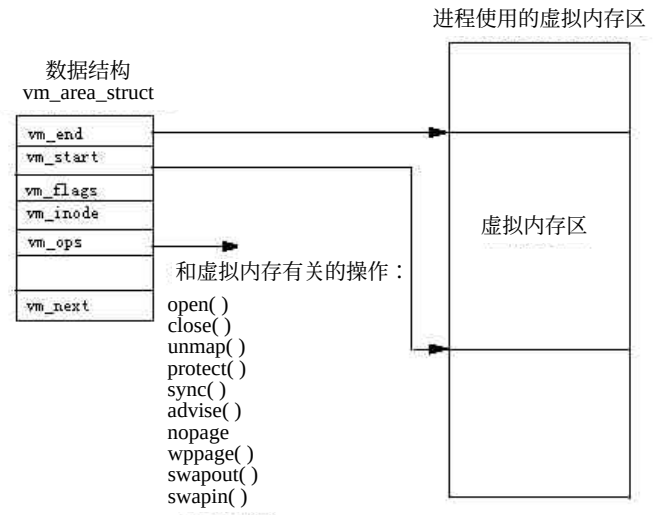

图10-4 虚拟内存数据结构示意图

链接的共享库也是一样要装入到虚拟内存空间。可执行文件并不是真正地装入到物理内存空间，它只是简单地链接到进程的虚拟内存。然后，随着应用程序运行时的需要，可执行镜像才逐渐地装入到物理内存。这种将一个文件的镜像和一个进程的虚拟内存地址空间连接起来的方法叫做内存映射。

数据结构mm_struct代表每个进程的虚拟内存空间。它包含了正在执行的镜像的信息和一些指向vm_area_struct结构的指针，如图10-4所示。每个vm_area_struct结构都描述了进程的虚拟内存的起始和结束位置，进程的存取权限以及和内存有关的一系列的操作。这些操作是Linux系统在处理虚拟内存时将要用到的。

当一个可执行镜像映射到一个进程的虚拟内存地址时，操作系统将创建一系列的数据结构vm_area_struct，每一个vm_area_struct代表可执行镜像的一部分。Linux系统支持多种标准虚拟内存操作，创建vm_area_struct时，相应的虚拟内存操作就会和vm_area_struct链接起来。

10.11 请求调页

一旦一个可执行镜像映射到了一个进程的虚拟内存中，它就可以开始执行了。因为开始时只有镜像开头的一小部分装入到了系统的物理内存中，所以不久进程就会存取一些不在物理内存中的虚拟内存页，这时处理器会通知 Linux发生了页面错误。页面错误将会描述页面错误发生时的虚拟内存地址和存取内存操作的类型。

Linux首先查找代表发生页面错误的虚拟内存区的 vm_area_struct。如果没有代表此出错虚拟地址的vm_area_struct，就说明进程存取了一个非法的虚拟内存地址。 Linux系统将向进程发出SIGSEGV信号。

Linux下一步检查页面错误的类型是否和此虚拟内存区所允许的操作类型相符。如果不符，Liniux系统也将报告内存错误。

如果Linux认为此页面错误是合法的，它将处理此页面错误。

Linux还必须区分页面是在交换文件中还是作为文件镜像的一部分存在于磁盘中。它靠检查出错页面的页面表来区分：

如果页面表的入口是无效的，但非空，说明页面在交换文件中。

最后，Linux调入所需的页面并更新进程的页面表。

10.12 页面高速缓存

Linux系统中页面高速缓存的作用是加快对磁盘中的文件的存取。对于已经作好了磁盘映射的文件，Linux每次读取一页，并将读取的页面存储到页面高速缓存中。

图10-5显示页面高速缓存由 page_hash_table组成， page_hash_table是一个包含指向mem_map_t结构指针的数组。

每当从一个内存映射文件中读取一个页面时，页面都要从页面高速缓存中读取。如果页面在高速缓存中，则将一个指向mem_map_t的指针返回给页面错误处理程序。否则，页面必须从磁盘上读入到内存中。

如果可能，Linux系统将会提前读取文件中的下一个页面，这样，如果文件是顺序执行的，那么下一个页面就已经在内存中了。

随着文件的读入和执行，页面高速缓存也将变得越来越大。不用的页面将被移出高速缓存。

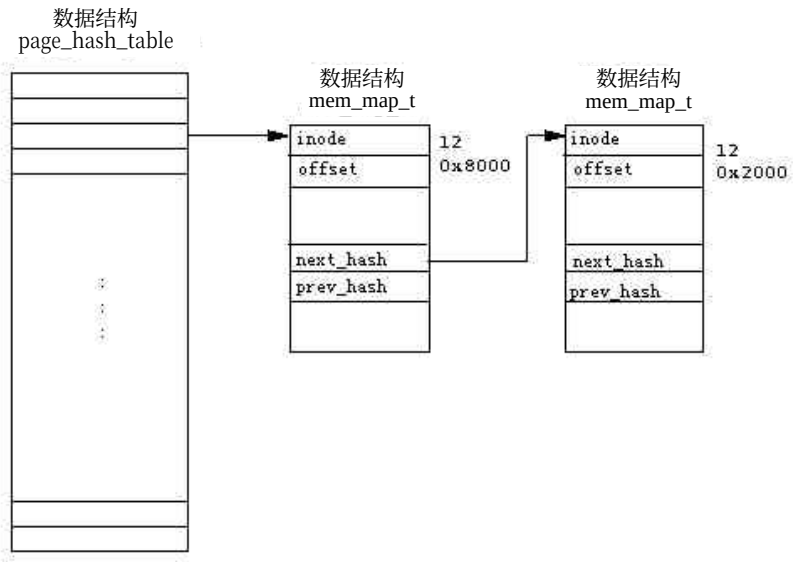

图10-5 页面缓存示意图

10.13 内核交换守护进程

当物理内存变少时，Linux内存管理必须释放物理内存页。此任务由内核中的交换守护进程(kswapd)完成。

交换守护进程是一个内核线程。内核线程是无需使用虚拟内存的进程，它们在物理内存中运行于内核方式下。交换守护进程还有一个重要的任务是保证系统中有足够的空闲内存。

交换守护进程是由内核的初始化进程启动的，并一直等候直到内核交换计时器周期性截止。

每当计时器截止时，交换守护进程都要检查系统中的空闲的页面数是否太少。它使用两个变量free_pages_high 和 free_pages_low来决定是否需要释放一些内存。只要空闲的内存数大于free_pages_high，交换守护进程就不做任何事，它将再一次进入睡眠状态直到计时器再一次截止。

检查系统中空闲页的目的是使交换守护进程可以计算出需要往交换文件中写入的页面数，此数目保存在nr_async_pages中。每次有页面排队准备写入到交换文件中时，nr_async_pages的值就会增加，而当写入结束后，nr_async_pages的值就将减少。free_pages_low 和 free_pages_high是在系统启动时设定的，并和系统的物理页面数有关。如果系统中的空闲页面数低于free_pages_high，或者甚至低于free_pages_low，则内核的交换守护进程将用以下三种办法减少正在使用的页面数：

- 减少缓冲区和页面高速缓存的大小。
- 把System V的共享内存页交换出系统内存。
- 交换或扔掉内存页。

如果系统中的空闲页面数低于free_pages_low，则内核交换守护进程将试图一次空出6个内存页，否则，内核交换守护进程将空出 3个内存页。上述的三种办法将轮流使用，直到空闲出足够的内存页为止。内核交换守护进程将会记录最后一次释放内存时使用的方法，下一次它再运行时，将会首先使用此方法。

在释放出足够的内存页之后，内核交换守护进程将进入睡眠状态，直到计时器截止。如果上一次的交换是由于内存空闲的页面数低于free_pages_low，则内核交换守护进程只睡眠一半的时间。