了更多的状态，但是却允许不同的内核操作来使用这些访问方法，这对进程而言更加高效。

10.4.4 Linux中的分页

早期的UNIX系统，每当所有的活动进程不能容纳在物理内存中时就用一个交换进程在内存和磁盘之间移动整个进程。Linux跟其他现代UNIX版本一样，不再移动整个进程了。内存管理单元是一个页，并且几乎所有的内存管理部件以页为操作粒度。交换子系统也是以页为操作粒度的，并且跟页框回收算法紧耦合在一起。这个后面会给予描述。

Linux分页背后的基本思想是简单的：为了运行，一个进程并不需要完全在内存中。实际上所需要的是用户结构和页表。如果这些被换进内存，那么进程被认为是“在内存中”，可以被调度运行了。代码、数据和栈段的页面是动态载入的，仅仅是在它们被引用的时候。如果用户结构和页表不在内存中，直到交换器把它们载入内存进程才能运行。

分页是一部分由内核实现而一部分由一个新的进程，页面守护进程，实现的。页面守护进程是进程2（进程0是idle进程，传统上称为交换器，而进程1是init，如图10-11所示）。跟所有守护进程一样，页面守护进程周期性地运行。一旦唤醒，它主动查找是否有工作要干。如果它发现空闲页面数量太少，就开始释放更多的页面。

Linux是一个请求换页系统，没有预分页和工作集的概念（尽管有个系统调用，其中用户可以给系统一个提示将要使用某个页面，希望需要的时候页面在内存中）。代码段和映射文件换页到它们各自在磁盘上的文件中。所有其他的都被换页到分页分区（如果存在）或者一个固定长度的分页文件，叫做交换区。分页文件可以被动态地添加或者删除，并且每个都有一个优先级。换页到一个独立的分区并且像一个原始设备那样访问的这种方式要比换页到一个文件的方式更加高效。有多个原因：首先，文件块和磁盘块的映射不需要了（节省了磁盘I/O读间接块）；其次，物理写可以是任意大小的，并不仅仅是文件块大小；第三，一个页总是被连续地写到磁盘，用一个分页文件，也许是或者也许不是这样的。

页面只有在需要的时候才在分页设备或者分区上被分配。每个设备和文件由一个位图开始说明哪些页面是空闲的。当一个没有备份存储的页面必须换出的时候，仍有空闲空间的最高优先级的分页分区或者文件被选中并且在其上面分配一个页面。正常情况下，分页分区（若存在）拥有比任何分页文件更高的优先级。页表被及时更新以反映页面已经不在内存了（如，page-not-present位被设置）同时磁盘位置被写入到页表项。

页面置换算法

页面替换是这样工作的。Linux试图保留一些空闲页面，这样可以在需要的时候分配它们。当然，这个页面池必须不断地加以补充。PFRA（页框回收算法）算法展示了它是如何发生的。

首先，Linux区分四种不同的页面：不可回收的（unreclaimable）、可交换的（swappable）、可同步的（syncable）、可丢弃的（discardable）。不可回收页面包括保留或者锁定页面、内核态栈等，不会被换出页面。可交换页必须在回收之前写回到交换区或者分页磁盘分区。可同步的页面如果被标记为dirty就必须要写回到磁盘。最后，可丢弃的页面可以被立即回收。

在启动的时候，init开启一个页面守护进程kswapd（每个内存节点都有一个），并且配置它们能周期性运行。每次kswapd被唤醒，它通过比较每个内存区域的高低水位来检查是否有足够的空闲页面可用。如果有足够的空闲页面，它就继续睡眠。当然它也可以在需要更多页面时被提前唤醒。如果任何内存区域的可用空间低于一个阈值，kswapd初始化页框回收算法。在每次运行过程中，仅有一个确定数目的页面被回收，典型值是32。这个值是受限的，以控制I/O压力（由PFRA操作导致的磁盘写的次数）。回收页面的数量和扫描页面的总数量是可配置的参数。

每次PFRA执行，它首先回收容易的页面，然后处理更难的。可丢弃页面和未被引用的页面都是可以被立即回收的，同时把它们添加到区域的空闲链表中。接着它查找有备份存储同时近期未被使用的页面，使用一个类似于时钟的算法。再后来就是用户使用不多的共享页面。共享页面带来的挑战是，如果一个页面被回收，那么所有共享了该页面的所有地址空间的页表都要同步更新。Linux维护高效的类树数据结构来方便地找到一个共享页面的所有使用者。普通用户页面在此之后被查找，如果被选中换出，它们必须被调度写入交换区。系统的swappiness，即有备份存储的页面和在PFRA中被换出的页面的比率，