第三部分　程序交换、基本输入/输出、块设备

第三部分说明主存和磁盘存储之间的基本输入/输出操作。

这些操作对于程序交换活动和磁盘文件的创建和引用都是基础性的。

本部分也介绍使用和处理大缓存(512字节缓存)的有关过程。

第14章　程序交换

与所有分时系统和某些多道程序系统一样，UN1X使用“程序交换”(program swapping)技术使多个进程共享有限的物理主存资源。“程序交换”有时也被称之为“滚进/滚出”(roll-in/roll-out)。

可以选择已被挂起的进程，将它们的数据段(包括“每个进程数据区”)写到磁盘上的“交换区”(swap area)中，这种操作称之为“换出”(swapped out)。

已被换出进程原来占用的内存区就可分配给其他进程使用，而这种进程非常可能是将从磁盘“交换区”中“换入”(swapped in)的。

与“换出”有关的大部分决策以及与“换入”有关的全部决策是由“sched”过程作出的。“sched”直接调用(2034)“swap”过程(5196)，由其处理具体的“换入”操作，而“sched”需要执行“换出”操作时则调用(2024)“xswap”(4368)过程。

非常熟悉UNIX更早版本的读者可能会想到，为了执行“换出”进程的操作，原来的“sched”直接调用的是“swap”而不是“xswap”。增加“xswap”过程的原因与实现共享“正文段”(sharable text segment)有关。为此还增加了另外一些过程，读者可以在“text.c”文件中找到这些过程。

为了实现正文段特性，究竟增加了多少代码呢?对此进行估计会使我们对UXIX的一个方面有进一步的了解。在“text.c”文件中包含有4个过程，它们是：“xswap”、“xalloc”、“xfree”和“xccdec”，这4个过程都涉及在文件“text.h”中说明的“text.h”结构数组。在“sysl.c”和“slp.c”文件中也增加了一些代码。

14.1　正文段

正文段是只包含“纯代码和数据”的段，而“纯代码和数据”的含量是在程序执行的全过程中不会有任何的改变。所以，在执行同一程序的多个进程之间可以共享正文段。

当系统中的很多用户同时执行同一程序，例如编辑程序或“shell”时，由于共享同一正文段就能大大节省所需的存储空间。

与正文段有关的信息必须存放在操作系统的中心部分，为此设计了“ text ”数组。共享正文段的每一个进程在其“ u.u_textp ”中存放一个指向相应“ text ”结构数组元素的指针。

正文段存放在代码文件的开始部分。共享某一正文段的进程开始执行时先使所需正文段的一个副本存放到“ 交换 ”区中。

在共享某一正文段的所有进程终止或改换映像后，释放该正文段所占用的资源。在内存中当前并无进程引用某正文段时，释放它所占用的内存资源。一般而言，只要当前不存在引用某正文段的进程，则释放它所占用的“ 交换 ”区。

共享某一正文段的进程数以及共享某一正文段其映像在内存中的进程数分别由“ x-count ”和“ x-ccount ”记录。例程“ xfree ”和“ xccdec ”分别对这两个计数器执行减 1操作，当这两个计数器值分别为 0时，它们也进行释放相应资源的操作。 (只要使进程换出或终止，就调用 xccdec；只要一个进程终止，就调用 xfree。)

14.2 sched(1940)

＃0进程执行“ sched ”。当该进程并不正等待它所要求的输入 /输出操作完成时，它的大部分时间因下列原因而处于等待状态：

状况1. runout

换出在磁盘交换区上的进程没有一个准备运行，所以无事可做。在“ newproc ”或“ expand ”调用“ wakeap ”或“ xswap ”时，这种情况可能发生变化。

状况2. runin

至少有一个换出进程并准备运行，但是它驻在磁盘交换区上还没有超过 3s以及/或者在内存中的所有进程都是活跃的或者没有一个驻内存时间超过 2s。随着时间的推移 (由clock度量)或对“ sleep ”的调用，这种情况可能改变。

当这两种情况中的任一种终止时：

1958：使处理机在优先级 6级运行，这样，时钟就不能中断处理机运行并更改“ p_time ”的值。然后搜寻准备运行并且已被换出至磁盘交换区最长时间的进程。

1966：如果没有这种进程，则保持状况 1

1976：搜寻具适当长度能存放数据段的内存区。如果具有正文段而且它当前不在内存中，则对该区增加正文段的长度。

1982：如果具有适当长度的可用内存区，则程序转移至“ found2 ”(2031)。(注意，此程序并不处理下述情况：内存中有足够空间可以存放正文段和数据段，但分开在两个连续地址空间中。对此段程序进行修改以利用这种可能性是否值得呢 ?)

1990：在内存寻找一个并非＃ 0或已锁定 (亦即，正在换出)的进程，并且其状态是“ SWAIT ”或“ SSTOP ”，但并非 SSLEEP，亦即，该进程正等待一低优先程度的事件，或者在跟踪中已经暂停 (参见第 13章)。如果找到这种进程，则转移至 2021行，将该进程映像换出。

注意，这种处理似乎不利于“ proc ”数组中下标值较小项对应的进程。

2003：如果要被换进的映像在磁盘交换区上当前一次的驻留时间少于 3s，则保持状况2。

2005：搜寻在内存中的各进程，该进程不是# 0进程或者已锁定的进程，其状态是“ SRUN ”或“ SSLEEP ”(亦即，准备运行，或者正等待高优先程度的事件)。如果这种进程有多个，则选其中此次在内存中时间最长的一个。

2013：如果要被换出的进程映像此次在内存中的时间少于 2s，则保持状况2。这里使用的常数“ 2 ”(以及在2003行的“ 3 ”)带有某种程度的任意性。由于某种原因， UNIX的编程者没有给这两个常数以符号常数名，这偏离了 UNIX的一般编程风格。

2022：对此进程设置“ 未装入内存 ”标志，然后调用“ xswap ”(4368)将其映像换出。

注意，因为被换出的进程不是当前运行进程，所以这里没有设置“ SSWAP ”标志。(参见1907，2286行。)

2032：若需要将正文段读入内存。注意，“ swap ”过程的参数是：

- 在磁盘交换区中的一个地址。
- 一个内存地址(单位：32字块)。
- 长度(要传送的32字块数)。
- 一个方向指示标志(B_READ＝＝1 表示“ 从磁盘到内存 ”)。

2042：换入数据段，然后……

2044：将磁盘交换区释放至可用区列表，记录内存地址，设置“ SLOAD ”标志以及清累计时间指示器。

14.3 xswap(4368)

4373：如果没有提供“ oldsize ”数据，则使用存放在“ u ”中的数据段当前长度。

4375：为进程数据段在磁盘交换区中找到一个区间。 (注意，磁盘交换区以 512字符块为单位进行分配。)

4378：调用“ xccdec ”(4490)(无条件地!)，其功能是使1个计数器值减1，该计数器值与该正文段相关，它表示驻内存进程中引用该正文段的数量。若此计数值减为 0，则释放该正文段占用的内存区，使其成为可用区。 (不需要将正文段换出，我们将会观察到，在磁盘交换区中已经有一个副本。)

4379：在将进程换出时，设置“ SLOCK ”标志。其作用是阻止“ sched ”将正在换出的进程再次选为需换出的进程。 (如果某个非sched例程，例如“ expand ”引发了“ 换出 ”，才可能发生这种情况。)

4382：除非“ xswap ”是由“ newproc ”调用的，否则释放内存映像。

4388：如果“ runout ”标志已设置，则“ sched ”正等待要被“ 换入 ”的进程映像，所以将它唤醒。

14.4 xalloc(4433)

当正启动执行一新程序时，“ exec ”调用“ xalloc ”(3130)，其作用是分配或连接至一正文段。参数“ ip ”是一指向代码文件“ 方式 ” (mode)的指针。在此调用时，“ u.u_arg〔1〕”包含以字节为单位的正文段长度。

4439：如果没有正文段，则立即返回。

4441：查看“ text ”数组，寻找未使用的项和该正文段的项。如果能找到后者，则执行记录操作，然后转移至“ out ”(4474)。

4452：安排将正文段复制至磁盘交换区。初始化未使用的正文项，然后在磁盘交换区中得到所需空间。

4459：更改该进程占用的空间，使其大得足以包含 ppda和正文段。

4460：在读代码文件之前，要设置用户态段寄存器，调用“ estabur ”是必须的。

4461：一个UNIX进程一次只能启动一个输入 /输出操作。因此可能将 i/o参数存放在“ u ”结构中的规定位置 —— “ u.u_count ”、“ u.u_offset 〔〕”和“ u.u_base ”。

4462：八进制值020(十进制数16)是该代码文件中的一个位移量。

4463：信息将从用户地址空间中的 0地址单元开始读入。

4464：将代码文件的正文段部分读入当前数据段。

4467：将数据段(减除ppda)写到为正文段保留的磁盘交换区中。

4473：缩小数据段 —— 它将被换出。

4475：“ sched ”总是在换入数据段之前先换入正文段，所以一旦数据段已在内存中，就无法将正文段调入内存。为此，若正文段不在内存中，则先将数据段“ 换出 ”至磁盘交换区中。

我们将会注意到，只要情况开始转向复杂，那么处理正文段的代码就会采取非常保守的操作方式。例如，当在“ text ”数组中无可用项时，即调用“ panic ”(4451)，这似乎是一种相当过份的反应。但是，对“ text ”数组空间宽容的处理策略与想做得更好一些的代码相比，可

t ”调用“ xfree ”(3233)，当一个进程要改换其映像时，

指针设置为“ NULL ”。
其结果值为0，则……
保存的……
文段映像。
将“ inode ”引用计数器值减 1，然后如果需要，则删除该

立 ”(sticky bit)的屏蔽码，UPM的CHMOD(I)中说明了“ 粘
没有一个进程引用此正文段，也将其副本保存在磁盘交换
。对于经常使用的程序，例如“ shell ”或编辑程序，这