Windows 平台内存精简编程

Minimum Memory Usage Programming for Windows

刘 玉 （廊坊师院 数学与信息科学学院 河北 固安 065505）

摘要：本文分析了 Windows 内存管理机制，介绍了在 32 位的 Microsoft Windows 2000 / XP 平台上开发具有最小内存占用的应用软件的方法。

关键词：内存管理机制 进程工作集 动态释放

1 最小化的启示

当用开发工具生成一个简单的 Windows 应用程序，使之不具有任何功能，只是一个程序框架。然后运行这个程序，这时我们调出 Windows 的任务管理器，在"进程"一栏观察该程序的内存占用，居然接近 7M 字节！

点击程序的"最小化"按钮，程序最小化后，当前程序的内存占用变为 752 K 个字节！如图 1。

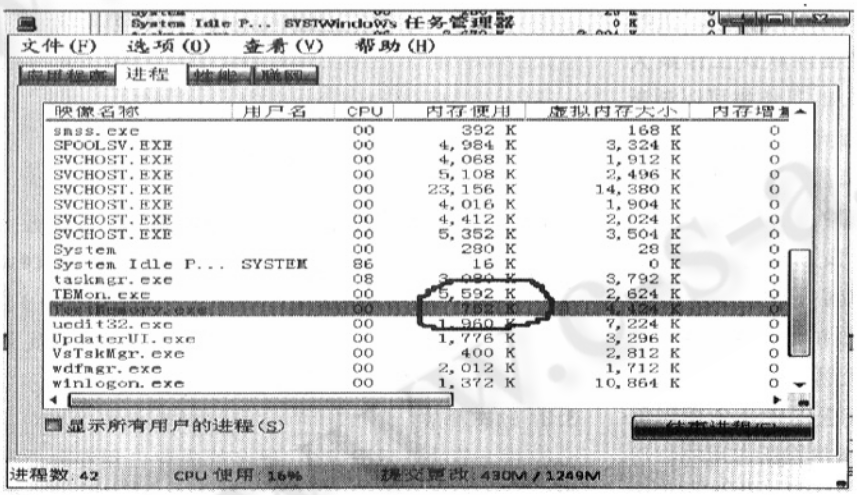

图 1 Windows 任务管理器最小化后进程显示

一个简单的窗口最小化，使该程序的内存占用精简了近十倍！在窗口最小化背后，Windows 系统究竟做了什么？我们是否也能将这项技术运用到实际的编程中？

2 简述 Windows 分页管理机制

32 位的 Windows 为每一个进程分配了 4G 字节的地址空间。一般的 PC 不可能拥有如此大的内存空间，因此 Microsoft 引入了"虚拟内存"的概念来实现 32 位的内存寻址，即 Windows 管理的内存由物理内存和"可交换文件"组成。"可交换文件"位于磁盘上，Windows 在运行一个程序时，将当前暂时不用的代码和数据转移到磁盘的可交换文件中，当需要这些数据和代码时再从可交换文件中将其读入物理内存。

Windows 对内存采用分页的方式进行管理，每一页的大小为 4K 字节。Windows 通过一级和二级页表来实现 32 位的内存寻址。虚拟内存中的数据不一定必须在物理内存中，如果一个页面的数据在磁盘上，Windows 就在对应的页表入口的标志位中做一个标记，这样访问到这个页面时就引发一个"页面错误"。Windows 一旦捕捉到页面错误，就将相应的页面从磁盘载入物理内存并再次尝试读取，这个过程对应用程序来说是透明的，应用程序无需关心要访问的数据是在物理内存里还是在磁盘上。

2.1 内存分配方式

Windows 应用程序使用"VirtualAlloc"函数分配内存块。"VirtualAlloc"分为两个步骤:保留和提交。"保留"是将虚拟地址做个标记表示预订了这个位置,于是其他的内存分配请求就不会被分配到已经被预定的位置上。"提交"的意思是实际准备开始使用这个内存块。

2.2 采用"懒惰"策略

既使提交了内存块,Windows 也并不会立即为这段地址初始化页表。因为虽然一段内存被提交,但是某些区域却可能从来不会使用,所以为这些地址创建页表完全是徒劳的。Windows 采取"懒惰策略"来分配内存,即直到某个页面错误出现,才为那个页面创建页表。懒惰策略极大地提高了 Windows 的内存分配性能。

2.3 创建进程工作集

Windows 为每一个进程创建工作集。即该进程最近 N 次访问物理内存的页面集合,N 被称为工作集窗口。

每当 Windows 启动时,根据当前计算机的内存量计算两个值:"进程默认工作集空间大小"和"进程最大工作集空间大小"。每个进程以默认工作集大小启动。随着进程使用内存的增加,工作集可以渐渐增大,直到最大值。如果系统有足够的空闲页面,进程工作集甚至可以超过最大值;如果系统没有多余的空闲页面,而进程又达到了最大工作集限制,对后续的页面错误,Windows 将先删除该进程的一个页面后,再将要求的页面载入。当空闲内存进一步减少时,Windows 将开始缩小各个进程的工作集,将一些页面换出内存。

进程工作集是 Windows 内存管理中一个相当重要的术语。进程工作集直接决定该进程所占用的物理内存页面的数目,所以可通过设置进程工作集大小的方式来控制当前进程的物理内存占用。

3 内存精简编程的实现

3.1 一个用来设置进程工作集的大小的 API 函数

BOOL SetProcessWorkingSetSize (HANDLE hProcess,

SIZE_T dwMinimumWorkingSetSize,

SIZE_T dwMaximumWorkingSetSize);

· 参数说明:"hProces"参数是进程句柄。"dwMinimumWorkingSetSize"和"dwMaximumWorkingSetSize"分别是最小和最大的的进程工作集空间大小(以字节为单位)。

· 返回值:操作成功与否的标志。

· 特别备注:当最小和最大的进程工作集空间大小同时被设置为"-1"时,Windows 将该进程的工作集大小设置为 0,并将进程移出物理内存。当该进程再发生任何 GUI 调用的时候,会立即产生"页面错误",这时 Windows 将重新为当前进程分配物理内存并将其调入内存执行。

3.2 演示程序

为验证上述 API 函数的执行效果,笔者编制了一个程序,界面如图 2。

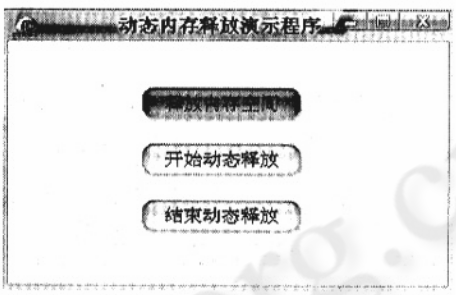

图 2 演示程序界

当点击"释放内存空间"按钮时,执行下段代码:

::SetProcessWorkingSetSize (::GetCurrentProcess ( ), -1, -1);// 设置当前进程工作集空间为 0

这时我们观察一下该程序的内存占用,会发现程序当前占用的物理内存变为不到 2M 字节!

3.3 动态释放

在上节的演示程序中,程序的内存占用变为小于 2M 字节,虽然比原内存占用小了很多,但是还未达到我们前面提到的程序最小化后(低于 1M 内存)的水平。

在我们释放了工作集空间后,任何在窗口内移动鼠标、响应窗口刷新等动作,都导致该进程发生 GUI 函数调用,这时 Windows 会重新为当前进程分配物理内存并将其调入执行。

所幸,可以采用"动态释放"的技术方案来解决这

个问题。我们可以创建一个会被周期性触发的函数，在这个函数里释放当前进程的工作集空间。对应于上节的演示程序，当我们点击了"开始动态释放"的按钮时，程序将创建一个以 500 毫秒为周期的时钟，当时钟被触发时，执行上节所列出的代码。这时我们再观察一下该程序的内存占用，您会发现程序当前占用的物理内存仅为 532K 字节！见图 3。

图 3　Windows 任务管理器下观察进程所占内存

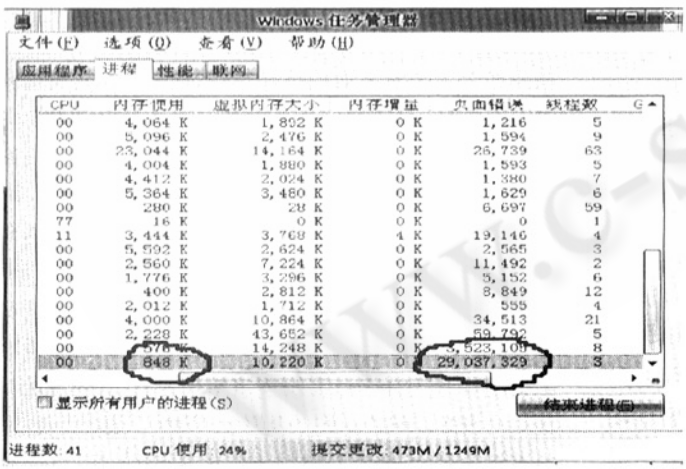

图 4　Windows 任务管理器下显示页面错误

3.4 寻求更好的解决方法

采用"动态释放"的方法虽然可为应用程序带来非常小的物理内存占用，但因为频繁地释放和申请内存，会在单位时间内产生大量的"页面错误"！每一个页面错误都会导致 Windows 与可交换文件进行一次交互，频繁的后台数据交换将占用大量的系统资源，降低 Windows 及其他当前应用程序的运行性能。下面是某著名软件的运行状况的截图，在图 4 中，虽然该软件占用的物理内存为 848K，但是其产生的页面错误竟然高达 29,037,329 个！

这是一个"最低内存占用"和"最佳运行性能"之间的矛盾。解决方案有如下两种：

(1) 摒弃周期性释放工作集的机制

通过取消周期性的释放操作，从根本上解决这个问题。同时防止程序进行 GUI 调用所导致的内存占用增大的问题。

一般具有"最低内存占用"需求的，往往是一些驻留在系统托盘的提供后台服务功能的程序。可以这样处理：每当程序隐藏至系统托盘后，执行一次内存释放；此后，直到程序被激活到前台之前，在每一次 GUI 调用之后，执行一次内存释放；一旦程序被激活到前台时，为了兼顾其自身的运行性能，不再进行内存释放操作。

(2) 优化释放周期

测试表明，毫秒级别的释放周期会带来严重的系统性能下降；当周期位于一秒与十秒之间时，系统性能则有了明显的回升；当周期上升为分钟以上级别时，用户将感觉不到内存的释放操作的运行开销。所以我们可以根据不同的应用情况，设置合理的时钟周期。

参考文献

1 Jeffrey Richter：Windows 核心编程[M]，机械工业出版社，2000.01。

2 Feng Yuan：Windows 图形编程[M]，清华大学出版社，2002.04。