**操作系统概念**

**调研报告**

**课程名称 操作系统**

**专业年级 信安21级**

**姓名学号 赵伯俣2021302181156**

**协 作 者 无**

**实验学期 2022-2023　 学年 第二 学期**

**课堂时数 32 课外时数**

**填写时间 2023 年 5 月 3 日**

一、Inter架构的内存管理机制32位/64位

1.总述

Inter体系结构的内存管理可分为两部分：分段和分页。

分段可以为每个程序或者任务提供单独的代码、数据和栈模块，可以使得多个进程或者作业能够在同一个处理器上运行而不互相干扰，从而保证了程序运行的效率。

分段机制将处理器可寻址空间分成若干个较小的受保护的地址空间段。段可以用来装载程序的代码，数据或者堆栈，也可以用来存放系统的数据结构。当处理器上运行多个进程时，操作系统为每个进程分配各自的段，同时处理器会规定这些段的边界，保证不同的进程之间在运行过程中不会相互干扰。

分页机制提供了虚拟内存系统，程序的执行代码按需要通过一定的方法被映射到物理内存中。该机制也可以用来隔离多个任务。

系统中所有的段都在处理器的线性地址空间内，但是只有逻辑地址（虚拟地址）才能确定一个数据单元（字节）在特定段中的位置。所以在实际程序访问内存时，需要有某种机制将逻辑地址映射到线性地址。逻辑地址由段选择符和偏移量组成。通过段选择符，可以在全局（或局部）描述符表中唯一定位一个段描述符，每个段描述符记录一个段在线性地址空间的基地址，然后通过和偏移量相加，得到实际的线性地址。即线性地址为基地址与偏移量的加和。

段选择符代表就是CS寄存器，是16位的。偏移量是EIP寄存器，是32位的。将逻辑地址映射到线性地址过程其实就是将CS中16位的逻辑地址映射成32位的线性地址。

2.使用分段机制所创建的模型

（1）基本平坦模型（Basic Flat Model）

该模型利用分段来保护程序。操作系统和应用程序可以访问一个连续的，没有分段的地址空间，即使所访问的地址处并没有物理内存时，也不会产生“超出内存范围”异常。

（2）受保护的平坦模型（Protected Flat Model）

该模型是在基本平坦模型的基础上将段限长被设定为实际物理内存范围内，如果试图访问实际内存以外的地址，会产生一个通用保护异常。

（3）多段模型（Multi-Segment Model）

该模型提供了对代码，数据结构以及程序的硬件级的强制保护。在这里，每个进程拥有自己的段描述符表及自己的段。

3.逻辑地址、线性地址和物理地址的转换

（1）逻辑地址

逻辑地址代表该进程所在段基地址的偏移地址，由应用程序直接使用的地址，而不需要关心分段分页机制转换的物理地址，也经常被称为虚拟地址。逻辑地址由16位的段选择符（CS）和一个32位的偏移量（EIP）组成。

（2）线性地址

作为逻辑地址和物理地址转换的一个中间值。逻辑地址是偏移量，段描述符中的地址是基地址，二者相加的结果是线性地址。线性地址是32位地址，地址空间中包含了所有的段以及为系统而定义的各种系统表。

（3）物理地址

实际内存中的地址，即地址总线上的寻址。根据是否采用分页机制分为两种情况，如果没有采用分页机制，那么线性地址就是物理地址；如果采用分页机制，那么需要通过页目录和页表转换成物理地址。32位机器总共有32根地址总线，实际物理地址内存空间为4GB，即2的32次方。

（4）三者的转换关系

线性地址是将段描述符中的段基址与偏移量相加所构成。如果没有采用分页，那么线性地址就是物理地址，否则，需要根据页目录和也表将线性地址转换成物理地址。

4.分段机制

分页机制提供了一种虚拟内存环境，虚拟内存通过一个较小的物理内存以及磁盘空间来模拟一个很大的线性地址空间。在分页机制下，每个段被分成很多页（通常一个页的大小是4KB），这些页或者在物理内存中，或者在磁盘上。操作系统会维护一个页目录和一组页表来跟踪这些页。当一个进程试图访问线性地址空间的一个地址时，处理器会通过页目录和页表将线性地址映射成物理地址，然后对其进行读写操作。如果被访问的页不在当前的物理内存中，会发生页故障，处理器会从磁盘上读取缺少的这页到物理内存中，接着执行操作。

处理器用来映射线性地址到物理地址的信息和产生缺少缺页异常的信息都在页目录和页表中，页目录和页表都储存在内存中。

为了减少地址转换所使用的总线周期，最近被访问过的页目录和页表项都被缓存在一个叫做转换后备缓冲区（TLBs）的设备中。TLBs可以满足多数的读当前页目录和页表的请求而不占用总线周期。仅当所访问的页表项不在TLBs中时，才需要额外的总线周期。

5.分页机制与分段机制的不同点

分页使用固定大小的页面，页有固定的尺寸。一个进程的数据结构可以部分在内存，部分在磁盘。

而分段的每个段的大小不固定，与它所持有的代码，数据总和有关。如果仅仅使用分段作为唯一的地址转换形式，一个进程的数据结构必须全部在物理内存中。

6.分页机制中所采用的页表和页目录

当启动分页机制时，处理器用来进行线性地址到物理地址转换的信息都包含在以下的4个数据结构中。

（1）页目录，一个由32位页目录项组成的数组。它被放在一个4KB的页中，页目录最多包含1024个页目录。

（2）页表，一个由32位页表项组成的数组。它存放在一个4KB的页中，页表最多包含1024个页表项。对于2MB和4MB的页，不适用页表。这些页直接从一个或者更多的页目录项映射。

（3）页，一个4KB，2MB或者4MB的平坦地址空间。

（4）页目录指针表，由4个64位的项组成的数组，每一项都指向一个页目录。仅当启用物理地址扩展时才使用这个数据结构。

二、以Windows为例，分析OS如何实现内存管理

1.虚拟内存与虚拟地址

每个进程创建加载的时候，会被分配一个大小为4G的连续的虚拟地址空间，实际上这个地址空间是不存在的，仅仅是每个进程“认为”自己拥有4G的内存，而实际上，它用了多少空间，操作系统就在磁盘上划出多少空间给它，等到进程真正运行的时候，需要某些数据并且数据不在物理内存中，才会触发缺页异常，进行数据转移和映射。

通过使用虚拟内存，程序可以使用一系列相邻的虚拟地址来访问物理内存中不相邻的大内存缓冲区。

不同进程使用的虚拟地址彼此隔离。一个进程中的代码无法更改正在由另一进程或操作系统使用的物理内存，这就是分段有助于共享和保护的好处。

程序可以使用一系列虚拟地址来访问大于可用物理内存的内存缓冲区。当物理内存的供应量变小时，内存管理器会将物理内存页（通常大小为 4 KB）保存到磁盘文件。数据或代码页会根据需要在物理内存与磁盘之间移动，比如4G的虚拟内存，实际上只有1G的物理内存，在操作系统也是可以实现的。

2.内存分页存储管理

把内存分为一个个相等的小分区,再按照分区大小把进程拆分成一个个小部分，将物理内存划分为若干固定大小的内存块， 称为物理页面或页框，虚拟地址划分为同样大小相同的块，称为页面。

通过分页，程序内存不需要全部加载到内存当中；在虚拟内存和物理内存建立映射关系后，通过虚存技术将不使用的页面放到磁盘中，当发生缺页中断的时候，再访问磁盘。

对于32位寻址的虚拟地址，如果页面大小为4KB，则虚拟页面数最多可以达到2^20个虚拟页面，那么页表的记录条数就为2^20条，所以需要使用多级页表。

在传统单表的页号部分继续划分分级页表。由一级页表中的信息可以映射到全部虚拟内存地址，但是计算内存占用，设每个页面4B, 一级页表：4KB， 二级页表：4KB \* 1KB = 4MB。

根据局部性原理， 虽然进程的虚拟内存很大，但是在程序运行过程中，并不会用到所有的虚拟地址，因此不需要全部记录在页表中。

为了提升虚拟地址到物理地址的转换速度会使用快表，快表可以理解为一种特殊的高速缓冲存储器（Cache），其中的内容是页表的一部分或者全部内容。作为页表的 Cache，它的作用与页表相似，但是提高了访问速率。由于采用页表做地址转换，读写内存数据时 CPU 要访问两次主存。有了快表，有时只要访问一次高速缓冲存储器，一次主存，这样可加速查找并提高指令执行速度。

使用快表之后的地址转换流程会相应的发生变化：首先根据虚拟地址中的页号查快表。如果该页在快表中，直接从快表中读取相应的物理地址；如果该页不在快表中，就访问内存中的页表，再从页表中得到物理地址，同时将页表中的该映射表项添加到快表中。当快表填满后，又要登记新页时，就按照一定的淘汰策略淘汰掉快表中的一个页。

3.段页式存储管理

分页和分段管理方式各有其优缺点，分页系统能有效提高内存的利用率，而分段则能更好地满足用户的需要，因此可以将两者结合成一种新的存储管理方式系统称为“段页式存储管理”即为先将逻辑空间按照段式管理分成若干段。再把段内空间按照页式管理等分成若干页。

4.页面置换算法

地址映射过程中，若在页面中发现所要访问的页面不在内存中，则发生页缺失中断。当发生缺页中断时，如果当前内存中并没有空闲的页面，操作系统就必须在内存选择一个页面将其移出内存，以便为即将调入的页面让出空间。用来选择淘汰哪一页的规则叫做页面置换算法。

（1）OPT 页面置换算法（最佳页面置换算法） ：最佳置换算法所选择的被淘汰页面将是以后永不使用的，或者是在最长时间内不再被访问的页面,这样可以保证获得最低的缺页率。但由于人们目前无法预知进程在内存下的若千页面中哪个是未来最长时间内不再被访问的，因而该算法无法实现。一般作为衡量其他置换算法的方法。

（2）FIFO 页面置换算法（先进先出页面置换算法） : 总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面进行淘汰。

（3）LRU 页面置换算法（最近最久未使用页面置换算法） ：LRU 算法赋予每个页面一个访问字段，用来记录一个页面自上次被访问以来所经历的时间 T，当须淘汰一个页面时，选择现有页面中其 T 值最大的，即最近最久未使用的页面予以淘汰。

（4）时钟页面置换算法（Clock）：需要用到页表项的访问位, 当一个页面被装入内存时, 把该位初始化为0。 然后如果这个页面被访问, 则把该位置设为1; 把各个页面组织成环形链表(类似钟表面), 把指针指向最老的页面;当发生一个缺页中断时, 考察指针所指向的最老页面, 若它的访问位为0, 立即淘汰; 若访问位为0, 然后指针往下移动一格。如此下去, 直到找到被淘汰的页面, 然后把指针移动到下一格。

（5）最不常用算法（LFU）：当一个缺页中断发生时, 选择访问次数最少的那个页面, 并淘汰。对每一个页面设置一个访问计数器, 每当一个页面被访问时, 该页面的访问计数器加1。 当发生缺页中断时, 淘汰计数值最小的那个页面。

5.内存分配

最简单的内存分配方法就是将内存分为多个固定大小的分区。每个分区可以只包含一个进程。所以多道程序的程度受限于分区数。

对于可变分区方法，操作系统维护一张表，用于记录哪些内存可用和哪些内存已用。开始时所有内存都可用于用户进程，因此可以作为一大块的可用内存，称为孔。最后内存有一个集合，以包含各种大小的孔。

通常可用的内存块为分散在内存里的不同大小的孔的集合。当新进程需要内存时，系统为该进程查找足够大的孔。如果孔太大，那么就分为两块：一块分给进程，一块返回孔集合。这种方法是通用动态存储分配问题的一个例子。从一组可用孔中选择一个空闲孔有以下常用的方法。

（1）首次适应：分配首个足够大的孔，查找可以从头开始，也可以从上次首次适应结束时开始。一旦找到足够大的空闲孔，就可以停止。

（2）最优适应：分配最小的足够大的孔。应查找整个列表，除非列表按大小排序。这种方法可以产生最小剩余孔。

（3）最差适应：分配最大的孔。同样应该查找整个列表，除非列表按大小排序这种方法可以产生最大剩余孔。

6.碎片

用于内存分配的首次适应和最优适应算法都会有外部碎片。对于内存的碎片可以是内部碎片，也可以是外部的，

内部碎片。通常按固定大小的块为单位来分配内存，采用这种方法，进程所分配的内存可能比所需要的大，这两个数字之差称为内部碎片，这部分内存存在于分区内部，但是又不能用。

外部碎片。外部碎片的一种解决方法是紧缩，移动内存内容，以便将所有的空闲空间合并成一整块。但是紧缩并不是总是可能的。如果重定位是静态的，并且在汇编时或加载时进行的，那么就不能紧缩；只有重定位是动态的，且在运行时进行的，那么才可以采用紧缩。

7.内存映射文件

内存映射文件与虚拟内存的不同之处在于给预定的地址空间区域调拨的物理存储器。虚拟内存调拨的物理存储器是内存或者来至系统的页交换文件；而内存映射文件调拨的是磁盘上的一个文件。

内存映射文件主要用在一下三个方面，即加载可执行文件、访问数据文件、多进程共享数据。

8.堆

堆是对虚拟内存的封装，可以使我们专注于解决实际业务问题，而不是考虑分配粒度、页面边界等问题。在系统内部，堆就是一块预定好的地址区域，随着不断的分配堆内存，堆管理器不断的调拨物理存储设备，释放堆内存也就释放已经调拨的存储器。