北京航空航天大学
Beijing University of Aeronautics and Astronautics

《操作系统》课程

第三章 存储器管理

授课教师：孙海龙

北航计算机学院OS教学组

2020年春季

内容提要

- 存储管理基础
- 页式内存管理
- 段式内存管理
- 虚拟内存管理
  - 局部性原理
  - 请求式分页
  - 页面置换
  - 关键设计问题
- 内存管理实例

常规存储管理的问题

常规存储管理方式的特征：

- 一次性：要求一个作业全部装入内存后方能运行。
- 驻留性：作业装入内存后一直驻留内存，直至结束。

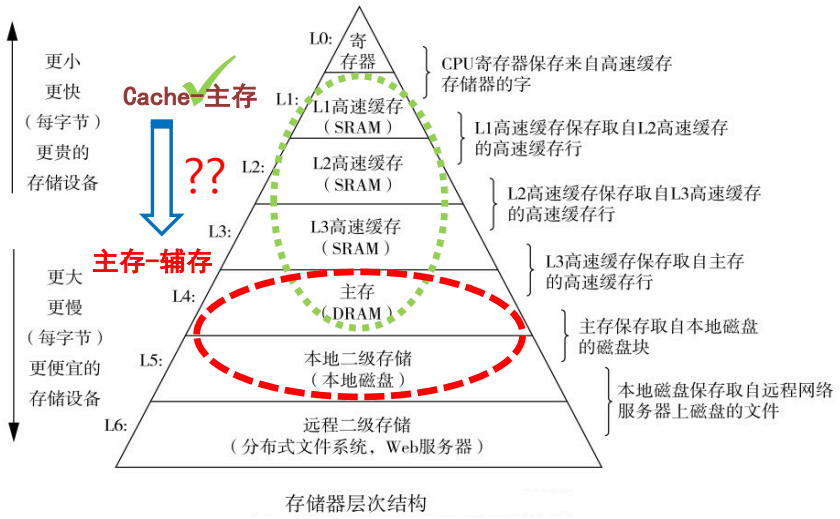
可能出现的问题：

- 有的作业很大，所需内存空间大于内存总容量，使作业无法运行。
- 有大量作业要求运行，但内存容量不足以容纳下所有作业，只能让一部分先运行，其它在外存等待。

已有解决方法：

- 增加内存容量
- 从逻辑上扩充内存容量：覆盖、对换。

再次回顾存储体系——借鉴已有方法



存储器层次结构

程序的局部性

- 程序在执行时，大部分是顺序执行的指令，少部分是转移和过程调用指令。(空间局部性)
- 过程调用的嵌套深度有限，因此执行的范围不超过这组嵌套的过程。（空间局部性）
- 程序中存在很多对特定数据结构的操作，如数组操作，往往局限在较小范围内。（空间局部性）
- 程序中存在很多循环结构，它们由少量指令组成，而被多次执行。（时间局部性）

局部性原理

- 指程序在执行过程中的一段较短时间内，所执行的指令地址和指令的操作数地址分别局限于一定区域。具体表现为：
  - 时间局部性：即一条指令的一次执行和下次执行，一个数据的一次访问和下次访问都集中在一段较短时间内；
  - 空间局部性：即当前指令和邻近的几条指令，当前访问的数据和邻近的数据都集中在一个较小区域内。

虚拟内存

虚拟内存是计算机系统存储管理的一种技术。它为每个进程提供了一个大的、一致的、连续可用的和私有的地址空间（一个连续完整的地址空间）。

虚拟存储提供了3个能力：

1. 给所有进程提供一致的地址空间，每个进程都认为自己是在独占使用单机系统的存储资源；
2. 保护每个进程的地址空间不被其他进程破坏，隔离了进程的地址访问；
3. 根据缓存原理，上层存储是下层存储的缓存，虚拟内存把主存作为磁盘的高速缓存，在主存和磁盘之间根据需要来回传送数据，高效地使用了主存；

虚拟存储管理的目标

- 借鉴覆盖技术：不必把程序的所有内容都放在内存中，因而能够运行比当前的空闲内存空间还要大的程序。
- 与覆盖不同：由操作系统自动完成，对程序员是透明的。

- 借鉴交换技术：能够实现进程在内存与外存之间的交换，因而获得更多的空闲内存空间。
- 与交换不同：只将进程的部分内容(更小的粒度，如分页)在内存和外存之间进行交换。

虚拟存储的基本原理

- 在程序装入时，不需要将其全部读入到内存，而只需将当前需要执行的部分页或段读入到内存，就可让程序开始执行。
- 在程序执行中，如果需执行的指令或访问的数据尚未在内存（称为缺页或缺段），则由处理器通知操作系统将相应的页或段调入到内存，然后继续执行程序。（请求调入功能）
- 操作系统将内存中暂时不使用的页或段调出保存在外存上，从而腾出空间存放将要装入的程序以及将要调入的页或段。（置换功能）

虚拟存储技术的特征

- 离散性：物理内存分配的不连续，虚拟地址空间使用的不连续（数据段和栈段之间的空闲空间，共享段和动态链接库占用的空间）
- 多次性：作业被分成多次调入内存运行。正是由于多次性，虚拟存储器才具备了逻辑上扩大内存的功能。多次性是虚拟存储器最重要的特征，其它任何存储器不具备这个特征。
- 对换性：允许在作业运行过程中进行换进、换出。换进、换出可提高内存利用率。

虚拟存储技术的特征（续）

- 虚拟性：允许程序从逻辑的角度访问存储器，而不考虑物理内存上可用的空间容量。
  - 范围大，但占用容量不超过物理内存和外存交换区容量之和。
  - 占用容量包括：进程地址空间中的各个段，操作系统代码。

虚拟性以多次性和对换性为基础，

多次性和对换性必须以离散分配为基础。

优点、代价和限制

优点：

- 可在较小的可用(物理)内存中执行较大的用户程序；
- 可在(物理)内存中容纳更多程序并发执行；
- 不必影响编程时的程序结构（与覆盖技术比较）
- 提供给用户可用的虚拟内存空间通常大于物理内存

代价：虚拟存储量的扩大是以牺牲 CPU 处理时间以及内外存交换时间为代价。

限制：虚拟内存的最大容量主要由计算机的地址结构决定。例如 32 位机器的虚拟存储器的最大容量就是 4GB。

与Cache-主存机制的异同

相同点：

1. 出发点相同：二者都是为了提高存储系统的性能价格比而构造的分层存储体系，都力图使存储系统的性能接近高速存储器，而价格和容量接近低速存储器。
2. 原理相同：都是利用了程序运行时的局部性原理把最近常用的信息块从相对慢速而大容量的存储器调入相对高速而小容量的存储器。

与Cache-主存机制的异同

不同点：

1. 侧重点不同：cache主要解决主存与CPU的速度差异问题；虚存主要解决存储容量问题，另外还包括存储管理、主存分配和存储保护等。
2. 数据通路不同：CPU与cache和主存之间均有直接访问通路，cache不命中时可直接访问主存；而虚存所依赖的辅存与CPU之间不存在直接的数据通路，当主存不命中时只能通过调页解决，CPU最终还是要访问主存。

与Cache-主存机制的异同

3. 透明性不同：cache的管理完全由硬件完成，对系统程序员和应用程序员均透明；而虚存管理由软件（OS）和硬件共同完成，由于软件的介入，虚存对实现存储管理的系统程序员不透明，而只对应用程序员透明（段式和段页式管理对应用程序员“半透明”）。
4. 未命中时的损失不同：由于主存的存取时间是cache的存取时间的5～10倍，而主存的存取速度通常比辅存的存取速度快上千倍，故主存未命中时系统的性能损失要远大于cache未命中时的损失。

实存（物理内存）管理与虚存管理

实存管理：

- 分区（Partitioning） （连续分配方式） （包括固定分区、可变分区）
- 分页（Paging）
- 分段（Segmentation）
- 段页式（Segmentation with paging）

虚存管理：

- 请求分页（Demand paging） - 主流技术
- 请求分段（Demand segmentation）
- 请求段页式（Demand SWP ）

请求分页（段）系统

- 在分页(段)系统的基础上，增加了请求调页(段)功能、页面(段)置换功能所形成的页(段)式虚拟存储器系统。
- 允许只装入若干页(段)的用户程序和数据，便可启动运行，以后在硬件支持下通过调页(段)功能和置换页(段)功能，陆续将要运行的页面(段)调入内存，同时把暂不运行的页面(段)换到外存上，置换时以页面(段)为单位。
- 系统须设置相应的硬件支持和软件：
  - 硬件支持：请求分页(段)的页(段)表机制、缺页(段)中断机构和地址变换机构。
  - 软件：请求调页(段)功能和页(段)置换功能的软件。

请求分页与分段系统的比较

	请求分页系统	请求分段系统
基本单位	页	段
长度	固定	可变
分配方式	固定分配	可变分配
复杂度	较简单	较复杂

虚存机制要解决的关键问题

1. 地址映射问题：进程空间到虚拟存储的映射问题；
2. 调入问题：决定哪些程序和数据应被调入主存，以及调入机制。
3. 替换问题：决定哪些程序和数据应被调出主存。
4. 更新问题：确保主存与辅存的一致性。
5. 其它问题：存储保护与程序重定位等问题

在操作系统的控制下，硬件和系统软件为用户解决了上述问题，从而使应用程序的编程大大简化。

请求分页式管理

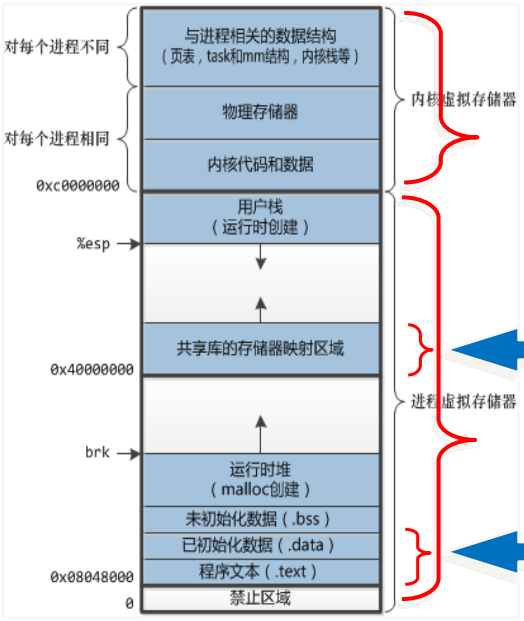
基本概念

1. 进程的逻辑空间（虚拟地址空间）

一个进程的逻辑空间的建立是通过链接器（Linker），将构成进程所需要的所有程序及运行所需环境，按照某种规则装配链接而形成的一种规范格式(布局)，这种格式按字节从0开始编址所形成的空间也称为该进程的逻辑地址空间。由于该逻辑空间并不是物理上真实存在的，所以也称为进程的虚拟地址空间。

如：Hello World进程包含Hello World可执行程序、printf函数（所在的）共享库程序以及OS相关程序。

进程的逻辑空间



用户不可见的进程空间，由OS使用

库文件(Printf函数）

用户可见的进程逻辑空间，如：程序、数据、堆和栈

Hello可执行文件

基本概念

2. 虚拟地址空间和虚拟存储空间

- 进程的虚拟地址空间即为进程在内存中存放的逻辑视图。因此，一个进程的虚拟地址空间的大小与该进程的虚拟存储空间的大小相同，都从0开始编址。

基本概念

3. 交换分区（交换文件）

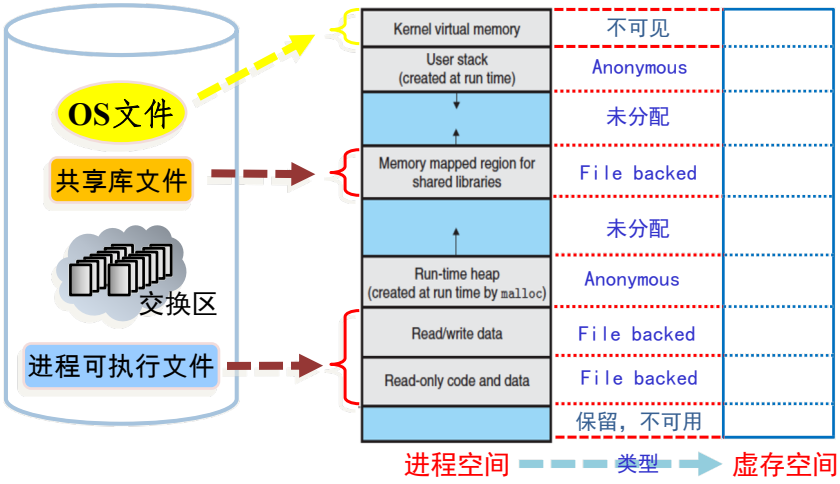
- 一段连续的磁盘空间（按页划分的），并且对用户不可见。它的功能就是在物理内存不够的情况下，操作系统先把内存中暂时不用的数据，存到硬盘的交换空间，腾出物理内存来让别的程序运行。
- 在Linux系统中，交换分区为Swap；在Windows系统中则以文件的形式存在（pagefile.sys）。
- 交换分区的大小：交换分区的大小应当与系统物理内存（M）的大小保持线性比例关系(Linux中）：

  If (M < 2GB) Swap = 2*M

  else Swap = M+2GB

地址映射问题（以32位Linux为例）

1. 进程空间到虚存空间的映射（进程的虚存分配）



地址映射问题

进程空间到虚存空间的映射（进程的虚存分配）

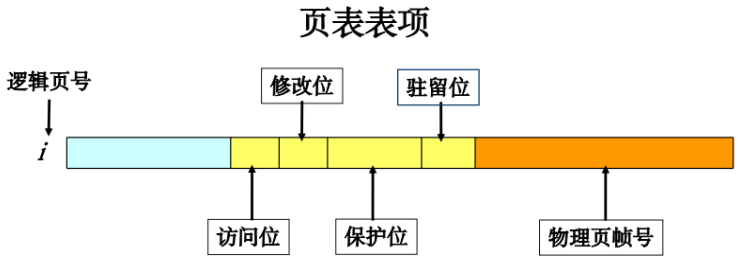
- 在程序装入时，由装载器（Loader）完成。
- 分配是以段为单位（需页对齐）进行的。
- 事实上，在每个进程创建加载时，内核只是为进程“创建”了虚拟内存的布局，同时建立好虚拟内存和磁盘文件之间的映射（即存储器映射）。实际上，并不立即就把虚拟内存对应位置的程序数据和代码（比如.text .data段）拷贝到物理内存中，等到运行对应的程序时，才会通过缺页异常，来拷贝数据。

地址映射问题

进程空间到虚存空间的映射（进程的虚存分配）

- 用户可执行文件（如Hello World可执行文件）及共享库（如HelloWorld中调用的库文件中的printf函数）都是以文件的形式存储在磁盘中，初始时其在页表中的类型为file backed，地址为相应文件的位置。
- 堆（heap）和栈（stack）在磁盘上没有对应的文件，页表中的类型为anonymous，地址为空。
- 未分配部分没有对应的页表项，只有在申请时（如使用malloc( )申请内存或用mmap( )将文件映射到用户空间）才建立相应的页表项。

请求式分页管理的页表



- 驻留位：1表示该页位于内存当中；0表示该页当前还在外存当中。
- 保护位：只读、可写、可执行。
- 修改位：表明此页在内存中是否被修改过。
- 访问（统计）位：用于页面置换算法。

Intel处理器的PDE和PTE

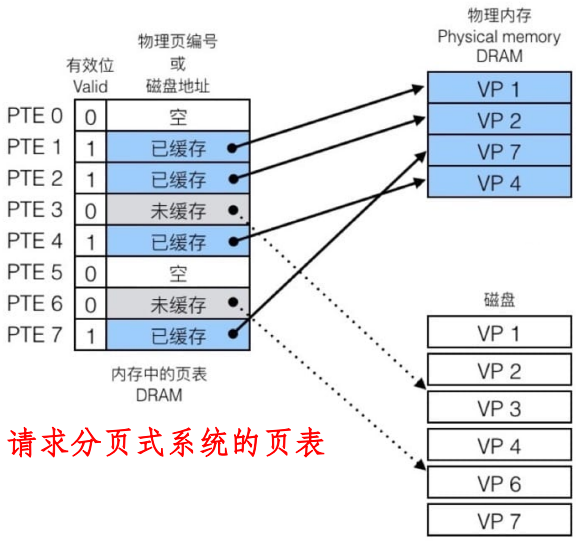
页目录项 PDE (Page Directory Entry)

PFN	Avail	G	PS	0	A	PCD	PWT	U/S	R/W	P

页表项 PTE (Page Table Entry)

PFN	Avail	G	0	D	A	PCD	PWT	U/S	R/W	P

PFN(Page Frame Number)：页框号
P(Present)：有效位
A(Accessed)：访问位
D(Dirty)：修改位
R/W(Read/Write)：只读/可读写
U/S(User/Supervisor)：用户/内核
PWT(Page Write Through)：缓存写策略
PCD(Page Cache Disable)：禁止缓存
PS(Page Size)：大页4M



请求分页式系统的页表