第 7 章 内存管理（多道程序设计下，将程序有效载入内存以让 CPU 执行）

关键术语：

页框：内存中固定长度的块

页：二级存储器中固定长度的数据块。数据页可以临时复制到内存的页框中。

段：二级存储器中变长数据块，整个段可以临时复制到内存的一个可用区域中(分段)，或可以将一个段分为许多页，然后将每页单独复制到内存中（段页式）。

7.1 内存管理的要求

1. 重定位（多道程序和共享内存技术要求程序使用**相对地址**以支持重定位）

动机：程序员不知道程序在运行时在内存中所处的位置；处于内存某一块区域的程序代码，在运行中可能被交换出(swap out)到外存，后来又被交换进(swap in)到内存的另外一块区域（**交换**技术）。

关注点：进程上下文与执行栈的位置与程序入口点 + 程序内内存访问（跳转访存等）。

解决：重定位一般要求一定的硬件（CPU）支持

2. 保护

目的：多道程序和共享内存技术要求一个进程不能对其他进程进行有意或无意的非授权访问。

问题：重定位导致无法在编译时检查绝对地址；

解决：多数程序设计语言允许地址的动态计算（如数组下标、指向某种数据结构的指针等）；通常整合在**重定位硬件机制**中(从软件上，难以预计所有非法情况，开销也太大)

3. 共享（灵活性）

含义：支持不同进程访问同一内存区域，包括：同一程序不同进程实例的共享代码段；合作进程间共享某些数据结构。**重定位机制**支持共享。

4. 逻辑组织

目的：将存储的一维线性结构模块化

模块化优势：独立编写；特权级设置，多个进程共享一个模块

5. 物理组织

含义：内存外存间的信息流组织，存储管理的本质，由 OS 负责。

覆盖技术（早期，复杂）：不同模块分配到同一内存区域，主程序负责按需换入换出。分为常驻段和覆盖段，要求用户给出代码段间的覆盖逻辑结构，只适用于单进程/作业。

交换技术（动态调度）：广泛用于小型分时系统，导致了虚存技术的出现。

7.2 内存分区

1.固定分区

含义：将内存划分成若干个等长/不等长的固定大小的区域

等长分区：进程大于分区只能部分载入，要应用覆盖技术；“小”进程将产生**内碎片**(internal fragmentation)（数据块小于分区大小），导致内存利用率降低。

放置算法：进程可以放到任意一个可用分区中；换出属于调度问题。

不等长分区：只能一定程度上减缓了等长分区存在的问题。

放置算法：

多队列：为每个分区设立一个输入队列，各个队列中的进程只能使用对应的分区，

评价：分区内减少了碎片，而某些队列为空时会造成内存空间的浪费（如小分区队列满而大分区队列空）

单队列：只有一个输入队列，进程使用可容纳它的最小空闲分区（无可用分区时可“交换”）

总结：相对简单，开销小；分区数目预设，限制了活动进程数；分区大小预设，小作业不能充分利用其占有空间。

2. 动态分区

含义：系统运行中分区数目和大小均可以改变。会导致**外部碎片**（分区外的存储空间变成碎片）

压缩：移动进程使相互紧靠（相当耗时，且需要进行动态重定位）

放置算法：首次适配；下次适配（通常分配位于内存末端的那块空闲区，致使大块空闲区很快被分裂，因此经常要压缩）；最佳适配（压缩更为频繁）

需要使用置换算法。

3. 伙伴系统——折中方案

4. 分区管理中的重定位

简单重定位：适用于多队列的不等长固定分区情况；只需要在进程第一次载入时把其中的内存引用全换成绝对地址

动态重定位：适用于等长固定分区，单队列的不等长固定分区；动态分区；“压缩”；基址寄存器；界限寄存器

7.3 分页

1. 概述

将主存划分成许多等长的小帧(frame，页框)；将进程划分成若干页(page)，一个页的大小与一个帧的大小相等。进程加载时，所有页面被载入可用帧（不要求连续），同时建立**页表**。

2. 页表

OS 为每个进程建立并维护一个页表

页表以页号为索引，每个表项包含该页在内存中对应的帧号（还包含保护、共享等信息）。OS 另外还维护一个空闲帧的列表。

3. 简单分页中的重定位

逻辑地址：与内存内容无关的内存位置

相对地址：相对于某一点（通常是程序起始位置，也可能是段基址）的逻辑地址（如偏移地址）

4. 简单分页的特点

页帧非常小，内碎片也小

分页中一个进程可占用多个页帧，从而不需要覆盖

分页中不要求一个进程占用的多个页帧连续

7.4 分段

1. 概述

将程序及数据划分成若干段(segment)（不要求等长，但不能超过最大长度）

2. 段表

段表以段号为索引，每个表项包含该段在内存中的起始物理地址、段长等。OS 另外还维护一个内存空闲块的列表。

3. 简单分段中的重定位

进程进入运行态时，其段表地址被载入 CPU 专用寄存器

4. 比较

分页是出于系统管理的需要，分段是出于用户应用的需要

分页是一维的，各个模块在链接时必须组织成同一个地址空间

分段是二维的，各个模块在链接时可以每个段组织成一个地址空间

分段方便实现模块化共享和保护，如程序可执行、数据可读写（段表表项要有保护位）

分段克服了分页存在的问题（数据结构的动态增长、动态链接、保护和共享）

分段存在外碎片，分页只有小的内碎片，分页内存利用率比分段高

第 8 章 虚拟内存（硬件支持+操作系统管理数据流）

虚存的特征

不连续性；部分交换；大空间；

硬件支持

内存管理硬件必须要支持分页/分段所需的动态地址转换等

软件支持

OS 必须管理内存与外存之间的页/段/段&页的交换

调页策略、放置策略、替换策略

驻留集和工作集管理

清除（回写）策略、加载（并发度）控制

部分加载的基础

逻辑地址动态转换 + 不连续分配（分页分段特点）

常驻集

程序执行的任何时候都在内存的部分

加载过程

若遇到一个逻辑地址访问不在内存中的块，则产生一个访问内存错误的中断。OS 响应中断时将进程置于阻塞态，并将逻辑地址访问的块读入内存。

结果

内存中可同时容纳更多的进程；进程可以比内存大，实现了**虚拟存储**

抖动

交换操作过于频繁。**局部性**原理保证了虚拟存储系统的可行性和效率性

8.1 硬件和控制结构

1. 虚拟页式存储管理

页表

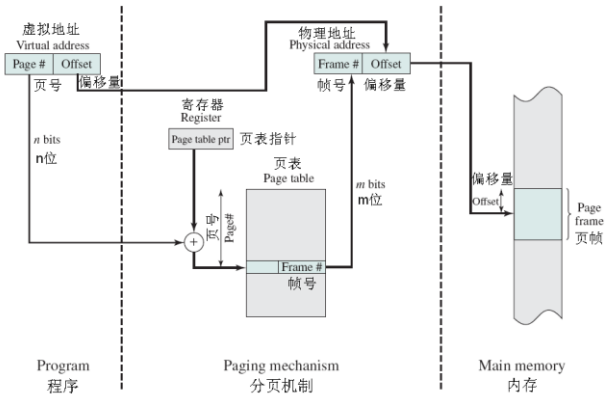
Present：在/不在内存

Modified：有没有被修改

Protection：保护码，1 位或多位(rwe：读/写/执行)

Referenced：有没有被访问

Cache：是否禁止缓存



Address Translation in a Paging System

分页系统中的地址转换

页表规模与问题

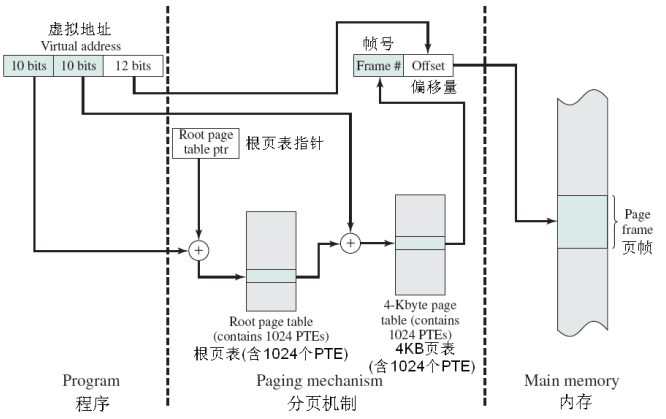
长度不定，取决于进程大小

起始地址保存在一个 CPU 专用寄存器里（Intel CPU 的为 CR3）

可能会非常大，从而占用内存也非常大。通常也对页表进行分页，存储到**虚存**。

多级页表（虚页号分多个域）

顶级页表在内存，其他级可在内外存间交换。



Address Translation in a Two-Level Paging System

二级分页系统中的地址转换

64 位处理器与**倒排页表**

动机：用于解决物理内存大大小于虚拟内存（CPU 的可寻址空间）的问题

实现：用一个链式 hash 表，虚拟地址的页号部分映射到倒排页表的某项的指针。这种方式将倒排页表的大小固定。

转换检测缓冲区 TLB（**联想存储器**）

动机：页表存储在内存致使每个内存引用至少要访问两次内存，大大影响效率

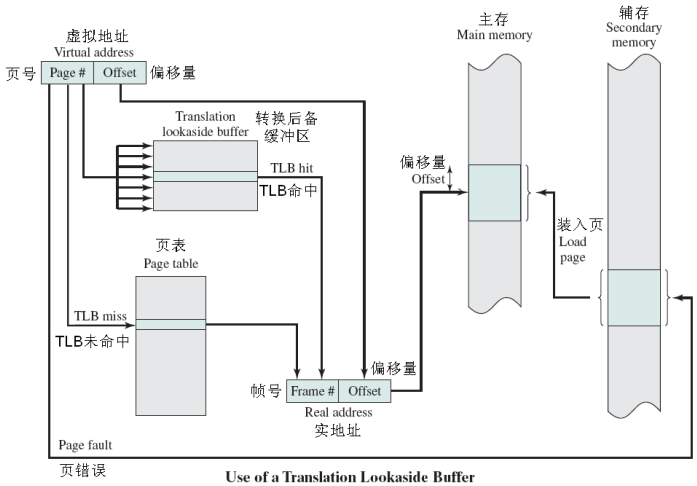
TLB 表项内容：有效位、虚页号、修改位、保护码、帧号等

工作原理：

给定一个逻辑地址，CPU 首先到 TLB 去检查，判断虚页号在不在其中（关联映射）

若在（命中，hit），则直接从 TLB 中提取帧号并形成物理地址

若不在（不中/未命中，miss），则按普通访问页表方式工作，形成物理地址，并更新 TLB（新找到的页表表项替换一个 TLB 表项）。



Use of a Translation Lookaside Buffer
转换后备缓冲区的用法

页面大小问题

小页面有利于减少内碎片总量

大页面有利于减小每进程的页表容量

大页面有利于实现有效的磁盘数据块传送

页面大小与缺页率

页面很小：每个进程的内存页较多，通过调页很快适应局部性原理的要求，缺页率低

页面很大：进程使用的大部分地址空间都在内存，缺页率低

页面中等大小：局部性区域只占每页的较小部分，缺页率高

页面大小与软件策略

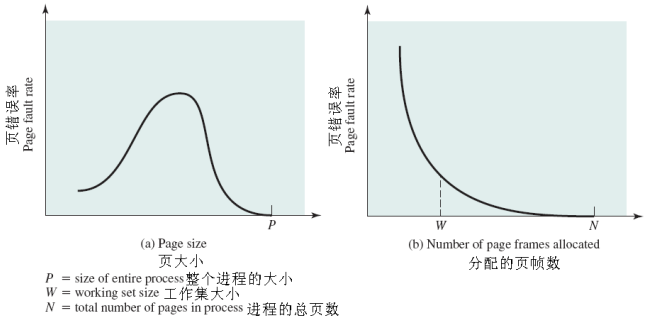
分配给进程的内存帧数可小于进程所需页面总数

页大小固定时：

数目越多，缺页率越低

页面数目的下限应该是一条指令及其操作数可能涉及的页面数

数目的上限应该是足以保证进程的每条指令都能被执行



Typical Paging Behavior of a Program
程序的典型分页行为

2. 虚拟段式存储管理

段式管理优点

简化了动态增长的数据结构的处理

支持模块的独立修改和重编译

更有效的进程共享

更容易实现保护

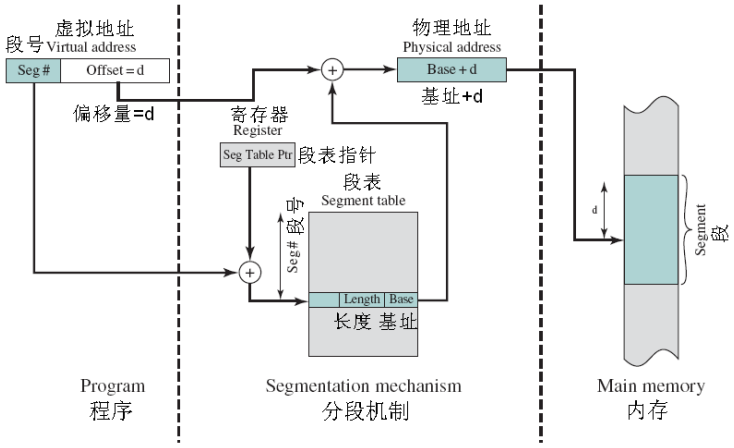
虚拟段式

段表：由表项组成，每个进程一张（LDT）

段表长度不定，存放在内存

当前进程的段表起始地址保存在 CPU 的一个专用寄存器里（如 Intel CPU 为 LDTR）

进程中有的段可能不在内存中（位于外存）



Address Translation in a Segmentation System

分段系统中的地址转换

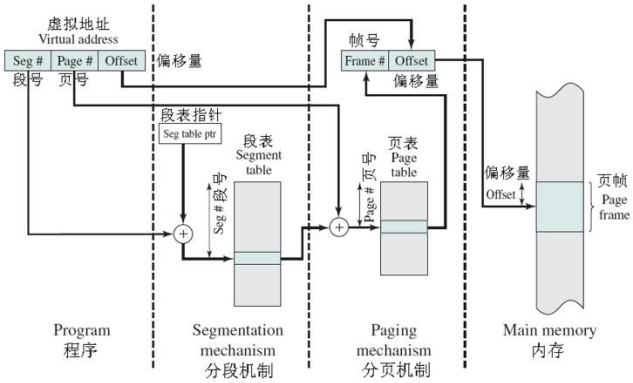
3. 虚拟段页式存储管理

基本原理：

将程序按逻辑结构划分成若干段，每个段进一步划分成若干个页

将内存划分成许多小的帧，帧与页大小相等

OS 为每个进程建立并维护一个段表，为每个段建立并维护一个页表



Address Translation in a Segmentation/Paging System

分段/分页系统中的地址转换

4. 共享与保护

分段提供了共享与保护。一个段可能会在多个进程的段表中引用。

环保护：

分层访问模式：

中心 0 号环为内核模式

内环 1 号环为执行模式

中环 2 号环为管理模式

外环 3 号环为用户模式

程序只能访问同层或更外层的数据

程序可以调用同层或更内层的服务

8.2 操作系统软件

目标：缺页率最小化

1. 调页策略

请求调页：只通过响应缺页中断调入需要的页面，也只调入发生缺页时所需的页面

预先调页：在发生缺页需要调入某页时，一次调入该页以及相邻的几个页

2. 放置策略

纯分段系统中，需要使用，类似动态分区，涉及外碎片、压缩操作等

最佳适配、首先适配

对纯页式或段页式系统不需要。

3. 替换策略

主要问题：

在计划置换的页集中，选择换出哪一页

不是所有内存中的页面都可以被替换：

页框锁定：OS 内核、关键控制结构、I/O 缓冲区等

驻留集策略决定了不同的替换范围：被替换的页面局限在本进程，或允许在其他进程。

最优算法

用作其他算法的性能评价依据

最近最少使用 LRU

淘汰内存中最久未使用的页面

性能接近最佳算法

需要记录页面使用时间的先后关系，实现开销太大（记录时间戳方法/页的访问栈方法）

先进先出

Belady 现象：在分页式虚拟存储器管理中，缺页置换算法采用 FIFO 算法时，如果对一个进程未分配它所要求的全部页面，有时就会出现分配的页面数增多但缺页率反而提高的异常现象。

	0	**1**	**2**	**3**	**0**	**1**	**4**	**0**	**1**	**2**	**3**	**4**
最年轻的页	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
最老的页			0	1	2	3	0	0	0	1	4	4
	F	F	F	F	F	F	F			F	F	

	0	**1**	**2**	**3**	**0**	**1**	**4**	**0**	**1**	**2**	**3**	**4**
最年轻的页	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
			0	1	1	1	2	3	4	0	1	2
最老的页				0	0	0	1	2	3	4	0	1
	F	F	F	F			F	F	F	F	F	F

时钟算法

环形链表实现算法

每页关联一个使用位。在页首次被装入时和发生缺页后被访问时，置 R 为 1

在替换算法扫描后，置 R 为 0

当需要置换页时，从指针所在的当前位置开始扫描整个缓冲区，选择遇到的第一个使用位为 0 的帧进行替换，替换后指向下一页。

4. 驻留集和工作集管理

主要问题：

给每个活动进程分配多少个页框

计划置换的页集是那些产生缺页中断的进程还是所有页框都在内存中的进程

术语：

驻留集(resident set)指虚拟页式存储管理中给进程分配的物理页面（帧）的集合；

驻留集大小即是这个集合的（帧）元素个数

驻留集大小与系统效率

每个进程的驻留集越小，则同时驻留内存的进程就越多，CPU 利用率越高

进程的驻留集太小的话，则缺页率高，使调页的开销增大

进程的驻留集大小达到一定数目之后，再给它分配更多页面，缺页率不再明显下降

固定分配策略

在执行过程中进程的驻留集大小固定

替换页面时从各自驻留集中选择

可变分配策略

在执行过程中进程的驻留集大小可变

根据缺页率动态调整，性能较好

需要 OS 对活动进程的行为进行评估，增加开销

替换范围

	全局替换	局部替换
固定分配	不可能	进程开始前需预先确定分配多少页面——多了会影响并发水平，少了会使缺页率过高
可变分配	最容易实现的组合，许多OS中采用（如Unix） 主要问题：如何决定哪个进程的页面将被替换，不利优化（“损人利己”）	试图克服全局替换的问题（具体做法见后） 可能是最佳组合（Windows NT采用）

固定+局部：缺点同固定

可变+全局：置换页的选择不能确定。

可变+全局：

进程加载进内存时，给它分配一定数目的物理页面；采用请求调页或预先调页填满这些物理页面

缺页时，从缺页进程本身的**驻留集**中选择替换一页面

定期重新评估进程的**驻留集**大小，并相应增加或减少，以提高系统整体性能

评估方法：工作集

工作集

一个进程执行过程中某段时间内所访问的页的集合，用 W(t, Δ)表示

利用工作集来进行驻留集调整的策略：

记录一个进程的工作集变化

定期删除**驻留集**中不在工作集中的页面

总是让**驻留集**包含工作集（不能包含时则增大**驻留集**）

存在问题：

工作集的过去变化未必能够预示工作集的将来(大小或组成页面均可能会改变)

记录每个进程的工作集变化所要求的开销太大

对工作集窗口大小 Δ 的最优值难以确定，而且通常该值是不断变化的

工作集接近策略

缺页率 (PFF)：跟踪缺页率而不是工作集的变化（在局部性阶段的过渡期间效果不好）

VSWS (可变间隔采样工作集)策略

通过增加采样频率来解决 PFF 算法的缺点

驱动参数：采样区间的最大/最小宽度 M/L（为异常条件提供边界保护）、采样实例间允许发生的缺页中断数 Q（使能正常激活采样）

策略：

采样间隔达到 L 时挂起进程并扫描使用位

若在采样间隔<L 时发生了 Q 次缺页中断

采样间隔<M，则一直等待

采样间隔≥M，则扫描使用位

5. 清除策略

请求清除：该页被置换之前才调出，即把清除推迟到最后一刻

问题：调入所缺页面之前还要调出已修改页面，缺页进程的等待时间较长

预先清除：该页被置换之前就调出，因而可以成批调出多个页面

若这批调出外存的页面中的多数在被置换之前还要被再次修改，则意义不大，形成不必要的开销

6. 负载控制

含义：决定内存中同时驻留的进程数目（即多道程序系统的并发水平）

太少，则通常所有进程可能都处于阻塞状态，从而 CPU 空闲时间太多

太多，则每个进程的驻留集太小，因此缺页频繁发生，导致“抖动”现象

策略一：基于**工作集**策略的算法（如缺页率 PFF 等）

它们隐含负载控制策略，只有那些驻留集足够大的进程才能运行，从而实现对负载的自动和动态控制

策略二：“**L = S** 判据”策略（ P.Denning，1980）

让缺页的平均间隔时间（是指真实时间而不是虚拟时间）等于对每次缺页的处理时间，研究表明这时 CPU 的利用率达到最大

一种类似的策略称为“50%判据”策略：让外存交换设备保持 50%利用率，这时 CPU 也达到最高的利用率

策略三：基于 Clock 替换算法的加载控制策略

定义一个轮转计数，描述轮转的速率（即扫描环形页面链的速率）

当轮转计数小于一定的阈值时，表明缺页较少或存在较多不常使用的页面，可提高系统负载

当轮转计数大于某阈值时，表明系统的进程并发水平过高，需降低系统负载

实现：当系统并发水平过高时（根据前述加载控制策略判定），需要降低系统负载

OS 不能完全控制进程的创建，但可通过进程挂起（中程调度）来减少驻留内存的进程数目。

即需要减少驻留内存的进程数目时，可以将部分进程挂起并全部换出到外存上。如低优先级的、缺页率高的、驻留集最小的、页面最多的，等等

第九章 单处理器调度

关键概念：

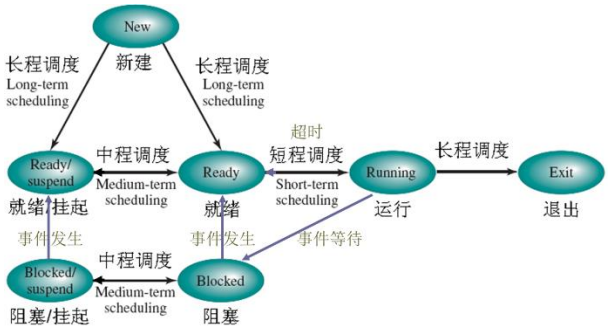
长程调度

中程调度

短程调度

I/O 调度（非处理器调度）

调度与**进程状态**转换关系



Scheduling and Process State Transitions

调度与进程状态转换

9.1 处理器调度类型

短程调度（导致当前进程阻塞或抢占当前运行进程的事件发生）

时机

当前进程正常或异常终止（通过中断实现）（？）

时钟或 I/O 中断

系统调用（通过软中断实现）

信号量操作（通过软中断实现）

模式

非剥夺式

剥夺式

过程

保存现场

根据某种调度算法选择下一个要运行的进程

如果没有就绪进程，系统会安排一个空闲进程(idle)，没有其他进程时该进程一直运行，执行过程中可接收中断

恢复现场

规则

面向用户+性能相关

周转时间（驻留时间）

响应时间

最后期限

面向用户+性能无关

可预测性

面向系统+性能相关

吞吐量

处理器利用率

面向系统+性能无关

公平性

强制优先级

平衡资源

9.2 进程调度算法

关键术语

决策模式（抢占非抢占）

归一化周转时间：周转时间/服务时间

类别 缩写	先来先服务 FCFS	轮转 RR	最短进程优先 SPN	最短剩余优先 SRT	最高响应比优先 HRRN	反馈 MF/FB
选择函数	max(w)	常数	min(s)	min(s-e)	max((w+s)/s)	e, 优先级
决策模式	非抢占	抢占 (时间片)	非抢占	抢占 (到达时)	非抢占	抢占 (时间片)
吞吐量	不强调	时间片太小时会低	高	高	高	不强调
响应时间	可能大	短进程小	短进程小	小	小	不强调
开销	最小	最小	可能高	可能高	可能高	可能高
对进程的影响	对短进程和I/O密集进程不利	公平对待	对长进程不利	对长进程不利	很好的平衡	可能对I/O密集进程有利
饥饿	无	无	可能	可能	无	可能

w：已等待时间、e：已执行时间、 s：进程所需总服务时间

先来先服务

不利于短进程，轮转/spn 解决

轮转

IO 低效，虚拟轮转解决

最短进程优先

可预测性低，指数平滑解决；长进程饥饿；缺少抢占机制，srt 解决

最短剩余时间

长进程饥饿

最高响应比优先

较为平衡

反馈法

长进程仍有可能饥饿；需要当一个进程在其队列中等待服务的时间超过一定时间后，提升他的优先级。

性能比较*

公平共享调度

基于进程组的调度：每组公平共享处理器时间

每个用户指定权值，表示其使用共享资源的份额

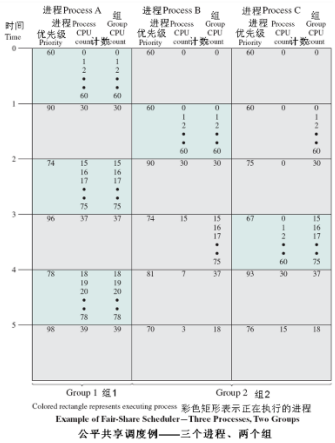
公平共享调度示例

进程j在时间段i开始处的优先级：

$P_j(i) = Base_j + CPU_j(i)/2 + GCPU_j(i)/4W_k$

其中：

- 进程j的基础优先级$Base_j$=60
- 进程j在时间段i中处理器使用计数$CPU_j(i)= CPU_j(i-1)/2$
- 组j在时间段i中处理器使用计数$GCPU_j(i)= GCPU_j(i-1)/2$
- 分配给组k的权值W_k=0.5



Example of Fair-Share Scheduler—Three Processes, Two Groups

公平共享调度例——三个进程、两个组

第 11 章 I/O 管理和磁盘调度

11.1 I/O 设备

分类

人可读的（human readable）——与用户交互

打印机

终端：显示器+键盘（+鼠标+触摸屏+手写笔+……）

机器可读（machine readable）——与电子设备通信

磁盘驱动器

固态硬盘/U 盘/闪存盘（Solid State Disk / USB key / USB flash drive）

传感器（sensor）

控制器（controller）

执行机构/驱动器/传动器（actuator）

通信（communication）——与远程设备通信

数字线路驱动器（digital line driver）

调制解调器（modem）

组成

机械部分：设备本身（物理装置）

电子部分：称作设备控制器（device controller）或适配器（adapter）

OS 和控制器打交道而不是设备本身

11.2 I/O 功能的组织

执行 IO 的三种方式：程序控制 IO，中断驱动方式，DMA

中断（数据传送由 CPU 完成）

操作系统将（带参数的） I/O 命令写入控制器寄存器

控制器从磁盘驱动器串行按位读块，直到将整块信息放入控制器的内部缓冲区中

控制器做和校验计算，以核实无读错误发生，然后控制器产生一个中断

CPU 响应中断，控制转给操作系统

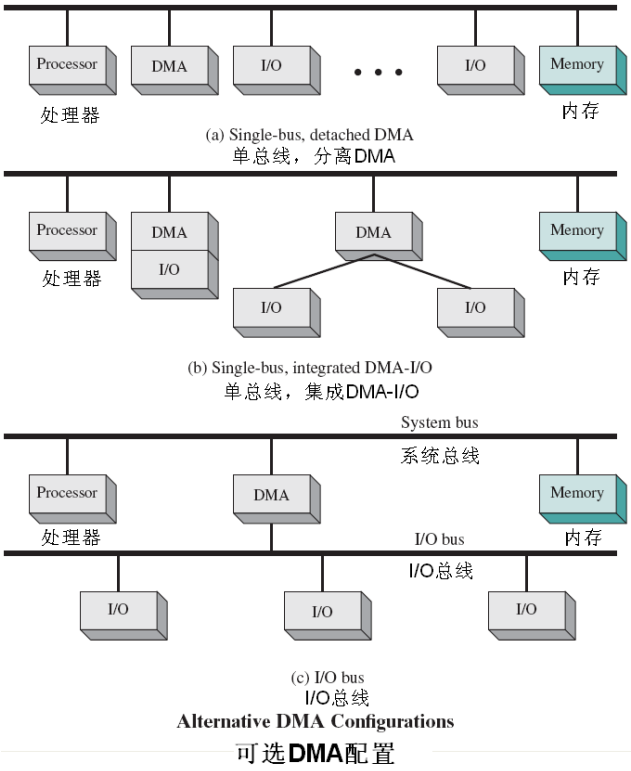
操作系统重复从控制器缓冲区中一次一个字节/字读这个磁盘块，并送入内存

DMA（数据传送不占 CPU 时间）

CPU 提供被读取块磁盘地址、目标存储地址、要读取的字节数

控制器将整块数据读进缓冲区，并进行核准校验
控制器按照指定存储器地址，把指定字节数数据送入主存
控制器引发中断，通知操作系统，操作完成

DMA 周期窃取技术（挂起一个指令周期）



Alternative DMA Configurations
可选DMA配置

11.3 OS 设计关注问题

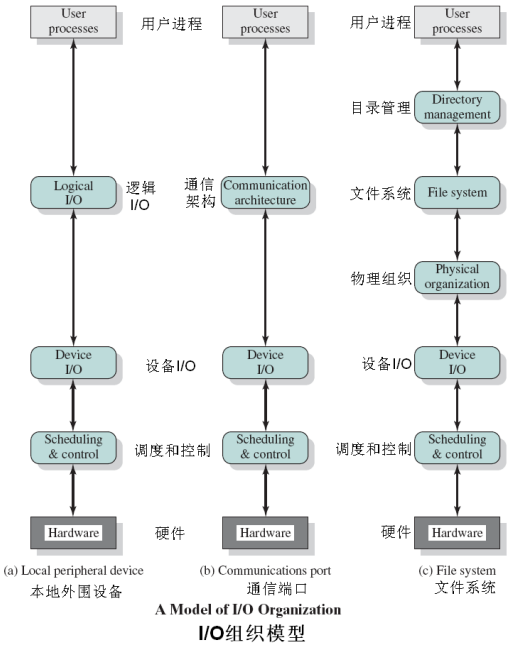
效率

IO 缓冲，磁盘调度，磁盘阵列，磁盘高速缓冲

通用性

模块化与层次结构

IO 功能的逻辑层次结构



A Model of I/O Organization
I/O组织模型

逻辑 IO

设备驱动程序的统一接口：实现一般设备都需的 I/O 功能，向用户层软件提供统一接口

设备 io

逻辑设备和物理设备间的过度协调；用于处理用户的 io 指令序列

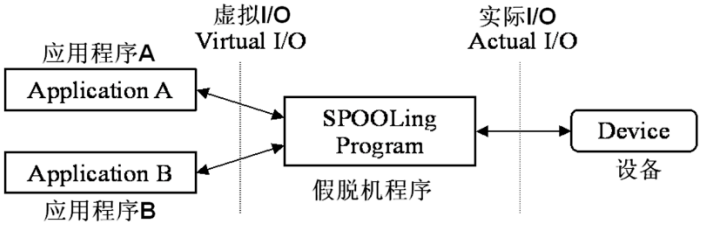
调度与控制

通常表现为设备驱动程序

用户空间的 I/O 软件*

库过程中的 I/O 系统调用

SPOOLing 系统：用共享设备(如磁盘)模拟独占设备(如打印机)



设置两级缓冲区：内存缓冲区和快速外存上的缓冲池，后者可以暂存多批 I/O 操作的较多数据

11.4 io 缓冲

原因（用户进程直接与 io 设备交互的问题）

进程必须要等待 I/O 操作完成才能继续（执行速度慢）

执行 I/O 操作期间一些页面必须保留在主存（干扰了 OS 的交换决策）

单进程死锁问题 p315

设备分类

面向块的设备（存储类）

面向流的设备（终端，打印机）

单缓冲

OS 为一个 I/O 请求在内存分配一个缓冲区

面向块的：

输入传送到缓冲区

需要时缓冲区中的块被移到用户空间

再移入另外一块到缓冲区——预读(read ahead)

面向流的：

每次传送一行/字节（缓冲区用来保存一行/字节的输入/输出）

用户每次输入一行/字节，并以回车表示一行/字节的结束

到终端的输出一次一行/字节

双缓冲（缓冲交换）

使用两个而不是一个系统缓冲区

OS 清空或填充一个缓冲区时，进程可以传送数据到另外一个缓冲区（或从另外一个缓冲区取数据）

循环缓冲

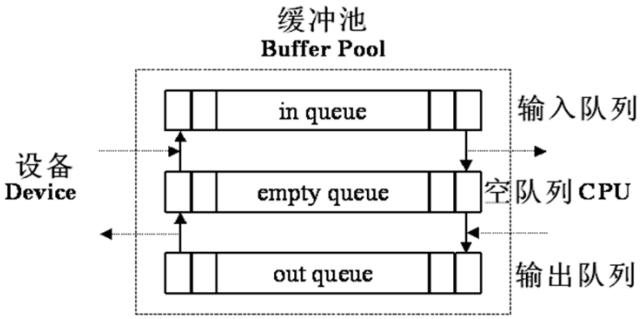
使用超过两个的缓冲区

每个缓冲区是循环缓冲区的一个单元

使 I/O 操作能跟上（可能爆发式地执行大量 I/O 的）进程的速度

有限缓冲区的生产者/消费者模型

缓冲池



11.5 磁盘调度

磁盘性能参数

寻道时间：磁头定位到磁道

寻道时间占一次磁盘读写时间的大部分

磁盘的一个重要特性：控制器是否支持重叠寻道（overlapped seeks），即是否可以同时控制两个或多个存取臂驱动器同时进行寻道

旋转延迟：磁头到达扇区开始位置

存取时间 = 寻道时间 + 旋转延迟

传输时间：数据传送部分时间

磁盘调度策略

基于请求者/队列属性

先进先出

效率不高，多个进程时性能接近随机调度

相邻两次请求可能会造成最内到最外的柱面寻道

优先级

短的批作业可能具有较高优先级，从而提供较小的交互响应时间

后进先出

减少磁臂运动，可能会有长进程饥饿

基于当前磁道位置

最短服务时间优先（寻道优先）

电梯算法（SCAN）：无访问请求，磁头不动；有请求按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务。

问题：一偏爱最靠里或最靠外的磁道的作业，二偏爱最近作业

单向扫描算法（ C-SCAN ）：从有读写请求的最低编号柱面开始递增扫描，途中按照柱面次序处理访问请求；处理完最高编号柱面上的请求后，存取臂立即带动读写磁头快速返回到有读写请求的最低编号柱面，返回时不处理任何访问请求；返回后可再次递增扫描。

克服 scan 的缺点一

N 步扫描算法（N-step-SCAN）

将磁盘请求队列分成若干个长度为 N 的子队列

每一次 SCAN 处理一个子队列

新来的请求必须加入到没在处理的子队列中

克服了 SCAN 缺点二

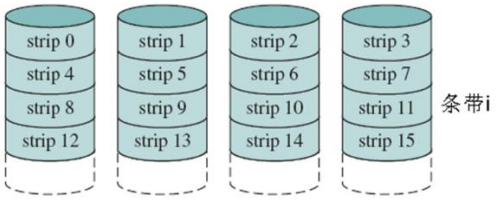
f-scan 其特例，使用两个子队列，开始时所有请求在一个队列中，另一个用来添加新请求

11.6 RAID

（凌好像只讲了 0，1）

RAID 0

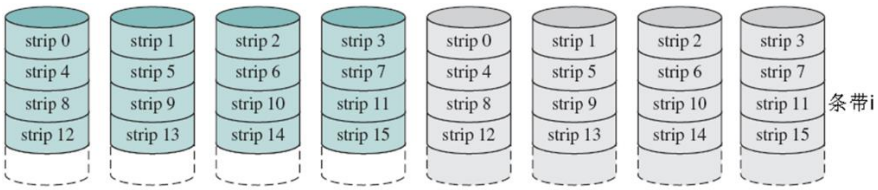
整个逻辑盘的数据被分散分布（通过阵列管理软件）在多个(至少两个)物理盘上，并行读写，无冗余能力



(a) RAID 0 (nonredundant)
无冗余

RAID 1

把一个磁盘的数据镜像到另一个磁盘上，可从任一磁盘读数据，数据同时更新到两个磁盘。磁盘利用率最低。



(b) RAID 1 (mirrored)
镜像

其他

类别	级别	说明	磁盘需求	数据可用性	大I/O数据量传输能力	小I/O请求率
条带	0	非冗余	N	低于单个磁盘	很高	读写都高
镜像	1	被镜像	2N	高于2~5、低于6	读高于0，写与0相近	读为0的2倍，写~0
并行访问	2	汉明码冗余	N+m	高于0可比	最高	0的2倍
	3	交错位奇偶	N+1			
独立访问	4	交错块奇偶	N+1		读与0相近，写（4明显）慢于0	读与0相近，写（4明显）慢于0
	5	交错块分布式奇偶校验	N+1			
	6	交错块双重分布式奇偶	N+2	最高	读与0相近，写慢于5	读与0相近，写明显慢于5

11.7 磁盘高速缓存

内存中的磁盘缓冲区，是磁盘中某些扇区的副本

磁盘缓冲替换算法

LRU, LFU, 基于频率的替换

第 12 章 文件管理

12.1 概述

文件结构：

域：基本数据单元（数据类型、长度，如姓名、年龄）

记录：一组相关的域（如雇员的信息）

文件：相似记录的集合（通过名字访问的实体）

数据库：相关数据的集合（由若干类型的文件组成）

文件管理的目的：

方便的文件访问和控制：以符号名作为文件标识，便于用户使用

并发文件访问和控制：在多道程系统中支持对文件的并发访问和控制

统一的用户接口：在不同设备（硬盘/U 盘/光盘）上提供同样的接口，方便用户操作和编程

多种文件访问权限：在多用户系统中的不同用户对同一文件会有不同的访问权限

优化性能：存储效率、检索性能、读写性能

差错恢复：能够验证文件的正确性，并具有一定的差错恢复能力

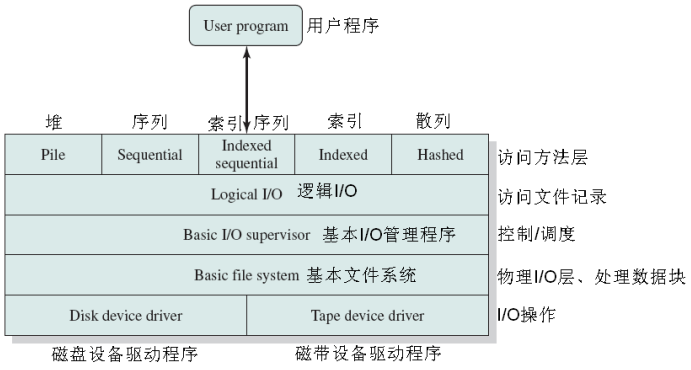
分类：

系统文件：有关 OS 及有关系统所组成的文件

库文件：标准子程序及常用应用程序组成的文件，允许用户使用但不能修改

用户文件

文件系统结构：



File System Software Architecture
文件系统软件架构

12.2 文件的组织与访问

文件组织：文件的逻辑结构

设计原则：

访问性能：访问快速（便于检索）、易于修改

存储性能：节省空间、维护简单（向物理存储转换方便）、可靠

常见逻辑结构：

	空间属性		修改记录大小		检索		
文件方法	可变	固定	相等	大于	单记录	子集	穷举
堆	A	B	A	E	E	D	B
顺序	F	A	D	F	F	D	A
索引顺序	F	B	B	D	B	D	B
索引	B	C	C	C	A	B	D
散列	F	B	B	F	B	F	E

A=O(r) 优秀
B=O(o x r) 良好
C=O(r logn) 尚可
D=O(n) 较差
E=O(r x n) 差
F=O(n^k>1) 不适合

O：复杂度
r：结果大小
o：溢出记录数
n：文件中的记录数

12.4 文件目录

文件说明（文件控制块，FCB，元数据）

操作系统为管理文件而设置的数据结构，存放了为管理文件所需的所有相关信息

目录

FCB 的有序集合，一个 FCB 就称作一个目录项，用于从文件名到文件内容的映射

通常将文件目录以文件的形式保存在外存，这个文件就叫目录文件（OS 所有）

内容（p350）

目录操作步骤

目录检索（根据用户给出的文件名，按名寻找目录项）

文件寻址（根据目录项中的文件物理地址等信息，计算出文件的任意记录或字节在存取介质上的地址）

目录操作类型

- 查找文件项（打开）
- 创建新文件
- 删除文件
- 列出目录中的文件
- 修改目录中的文件

目录结构

- 一级目录
  - 线性结构（检索时间长，命名冲突）
- 二级目录
  - 主文件目录（MFD）：用户名->子目录指针
  - 用户目录：一级目录
  - 缩短检索时间，部分克服命名冲突问题
  - 不利于用户对文件分管理
- 多级目录（层次结构）
  - 目录名：可修改
  - 目录树：中间节点是目录，叶子节点是目录或文件
  - 层次结构清晰，便于管理和保护，适用于较大的文件系统管理
  - 解决了命名冲突问题
  - 目录级别太多时，会增加路径检索时间（目录文件存放在外存）
- 改进多级目录
  - 将 FCB（目录项）分为两个部分
    - 符号目录项：文件名，文件内部标志号
    - 基本目录项：其他
    - 把符号目录项构成的符号文件目录组织成树状结构，按文件名排序
    - 把基本目录项构成的基本文件目录组织成线性结构,按文件内部标识
  - 排序（也称索引节点目录）