Computer system can be divided into four components

Hardware

Operating system

Application programs

Users （可以是人、机器或其他电脑）

Coordinate：协调

User操作系统是用户与计算机硬件之间接口

命令接口：联机命令/脱机命令；像鼠标键盘命令之类。

程序接口：系统调用组成，也称为广义指令。

System操作系统是计算机系统资源的管理者

The one program running at all times on the computer” is the kernel（内核）.

操作系统的一般定义：操作系统是一组有效控制和管理计算机各种硬件和软件资源，合理的组织计算机的工作流程，以及方便用户的程序的集合。

有效，合理（公平），方便

总线型、主机型（以存储器为核心）

硬件工作：

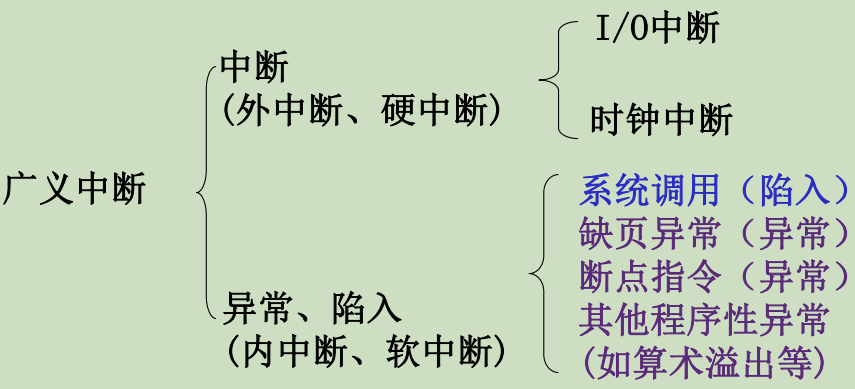
Each device controller is in charge of a particular device type（器件管理对映一种器件类别）

 Each device controller has a local buffer（都有一个local buffer）

 Each device controller type has an operating system device driver to manage it（器件管理都有一个操作系统device driver相对应）

操作系统是中断driven的

Trap is a software-generated interrupt

中断处理：

轮询：polling

vectored interrupt system 矢量化中断系统

Secondary storage（辅存）

Hard Disk Drives (HDD，硬盘）

Non-volatile memory (NVM，非易失存储器)

I/O方式

程序I/O (Programmed I/O)：循环检查方式控制

中断 I/O (Interrupt I/O)：同步I/O和异步I/O；完成I/O后，中断获取CPU算力，CPU需要在每个指令周期末尾检查中断。（每个数据的确认）

DMA方式：操作系统发出命令后，device直接和memory通信，全部结束后返回CPU完成信息。（一批数据的确认，CPU控制传输数据块大小及内存位置等信息）

通道方式（处理机，属于一种硬件？），CPU只需发出一条控制指令即可，通道会接管后续所有操作。（相关传递的信息，一个通道可控制多台设备/内存进行数据交换）

同步I/O：在I/O完成后再返回用户程序，一般同一时间只支持一个I/O操作

异步I/O：I/O操作不完成时，已返回用户程序

计算机体系架构：

单处理机系统：只有一个处理机

多处理机系统（使用效率上升，规模内经济下降，可靠性提高 - 容错上升）

多个处理机，每个处理器包含CPU，cache，register，但共用一个memory

同步多处理机系统；

异步多处理机系统；

Processor—A physical chip that contains one or more CPUs. （processor可有多个CPU）

 CPU—The hardware that executes instructions.

 Core—The basic computation unit of the CPU. （core与CPU对映）

 Multicore—Including multiple computing cores on the same CPU. （一个core上有多个计算核心）

 Multiprocessor—Including multiple processors. （多个CPU/core计算单元）

NUMA 非统一内存访问：内存访问时间取决于内存相对于处理器的位置。

被共享的内存物理上是分布式的，所有这些内存的集合就是全局地址空间。显然访问本地内存的速度要比访问全局共享内存或远程访问外地内存要快些。

UMA 统一内存访问：多个处理器通过统一总线访问存储器，每个处理器对内存的访问都是一致的。

缺点：任何数据都经过总线来进行，导致当CPU核心增加且总线带宽有限的情况下，内存的访问延迟增大。

集群系统（cluster）：是由一组互联的主机构成统一的计算机资源，通过相应软件协调工作的计算机机群，给人以一台机器的感觉。

Via storage-area network (SAN) 共享存储内容。

distributed lock manager (DLM，分布式锁管理器) 用于管理/避免运行操作冲突

对称集群/非对称集群（含有hot-standby mode监控与备份运行）。

内核态：又称kernel mode，monitor mode，system mode，supervisor mode，privileged mode

Reclaim 回收

多线程处理中每一个线程都有一个PC计数；

Suspend：结束

Resume：恢复

Mass-Storage：大容量

System security：可以使用user ID，group ID以及Privilege escalation（特权升级以暂时获得更多的权限）

Heterogeneous：异构的

批处理系统（batch system）（脱机使用） --- 程序

单道批处理：依次处理程序（高速CPU等低速I/O设备）

多道批处理：并发处理多道程序，内存同时存放多道程序（人机交互能力较弱）

分时操作系统：分时是指多个用户分时共享使用同一台计算机。 --- 任务

实时操作系统

网络操作系统（资源共享与通信）

分布式操作系统（多台机器协作完成同一任务）

响应时间：用户发出终端命令到系统作出响应的时间间隔。

设分时系统中进程数(用户数)为n，每个进程的运行时间片为q，则系统的响应时间为 S＝ n × q 。

Software as a Service (SaaS，软件即服务) – one or more applications available via the Internet。（软件应用服务）

Platform as a Service (PaaS，平台即服务) – software stack ready for application use via the Internet（可以自己部署软件的平台服务）

Infrastructure as a Service (IaaS，基础设施即服务) – servers or storage available over Internet（提供机器、存储和网络资源）

Precaution：预防措施

Command-Line Interface (CLI) ，命令行用户接口，文本界面

Graphics User Interface (GUI), 图形用户接口

Icon：图标

System call的传参方法：

寄存器传递参数（传递参数个数/长度受限）

地址传递参数

堆栈传递参数

Attach：连接

Detach：断开

Pertain：适用

不同的操作系统使用不同的system calls，所以各个操作系统有其使用范围

API定义了源代码和库之间的接口，同样的代码可以在支持这个API的任何系统中编译；

ABI允许编译好的二进制代码在使用兼容ABI的系统中无需改动就能运行。（描述了应用程序与操作系统、应用与库、各应用的组成部分之间的接口）

Policy策略: What will be done? （实现什么）

Mechanism机制: How to do it? （怎么实现）

操作系统结构

Simple结构（MS-DOS），用最小的空间提供最多的功能，接口与功能没有很好的分离。

分层结构，最高层为用户接口，只有单向依赖关系，并与调试与验证且易于扩展与维护。但是层际功能定义困难，且效率较低。

模块化，模块接口法。易于维护且有较强适应性，但无法决定验证调试顺序。（mac，linux有相关实现使用）

内聚性/耦合度，内聚性越高，耦合性越低，独立性越好。

宏内核（Monolithic Kernels），大内核，内核处理所有系统主要模块功能。有很高的使用效率，但难以发现error与bug，难以调试且内核会愈趋变大。（传统unix，linux也有）

微内核，微内核 - 服务器架构，内核只含有核心功能，只有微内核运作在内核态。其他操作系统功能在微内核外进程中实现，单个模块的崩溃不会影响内核运行。（windows，鸿蒙，mac都有）

一般包括进程管理，低级存储器管理，中断和陷入处理等功能。机制与策略相分离。易扩展、可靠安全且分布式计算。通信开销大，因而效率不高。

外核，为虚拟机分配资源，运行在内核态之中。一个硬件对映多个外核，一个外核对映一个操作系统。需要进行硬盘的重映射。

混合系统

系统启动

启动boot程序，boot程序往往放置于ROM区域中。ROM中仅有很小的一部分，完整的功能引导程序保存于磁盘的启动块上。Boot程序初始化CPU、寄存器等，接着启动操作系统，找到操作系统内核并加载运行。

具有启动分区的磁盘称为启动磁盘或系统磁盘。

激活CPU --- ROM中boot程序，指令寄存器置为BIOS --- 硬件自检 --- BIOS开始读取boot sequence--- 0号扇区的引导代码，MBR主引导记录 --- MBR含有磁盘分区表与标志 --- 寻找到引导分区，第一个扇区（引导扇区）--- 加载分区引导记录PBR，活动分区的第一个扇区 --- 启动管理器 --- 加载操作系统

应用故障 --- core dump file

操作系统故障 -- crash dump file

进程是一个具有一定独立功能的程序在一个数据集合上的一次动态执行过程。（数据+程序+进程控制块）

进程拥有代码段（text）、数据段（data，初始化与未初始化数据）、stack（临时变量）、heap（动态分配的数据）、寄存器与PC。

进程状态，运行、就绪、阻塞、终止、新（创立）

多次执行：一个程序可以对应多个进程

调用关系：一个进程可以包括多个程序

进程控制块（process control block）：进程序号信息、状态state、PC、寄存器、调度信息

Long term scheduler（job scheduler）长程调度，调度最长的（高级调度，作业调度）每个作业仅能调入一次与调出一次

Short term scheduler（CPU scheduler）短程调度（进程调度，低级调度）

Medium-term schedule中程调度（中级调度，内存调度）暂时不运行的进程放置于外存等待

I/O型进程，多I/O操作

CPU型进程，多CPU计算操作

Context switch依赖于硬件实现

父子进程的资源可以share，no share，share subset；父进程需要等待子进程结束且父子进程可以并行运行。

对父子进程共享数据需要修改时，产生拷贝 ---> 建立/申请新的空间并对应。

对于fork函数而言，父进程返回子进程进程号，子进程返回0

子进程

Zombie 僵尸进程，子进程运行终止，但父进程未调用wait()函数

Orphan 孤立进程，父进程没有调用wait()函数就停止

进程通信：interprocess communication（IPC）

Shared memory 共享内存

Message passing 消息传递（不借助共享变量）

需要建立连接后使用 send receive 通信

直接通信：进程间直接通信（link往往自动连接，且多为双向连接，each pair each link）

共享存储区：把一段物理内存映射到多个进程的内存空间（可能需要一定的数据结构控制buffer大小）

间接通信：进程间通过内核（信箱）进行通信（一对多，多对一）

Blocking 同步，sender block直到receive message

Non-blocking 异步，正好相反

信号，先到内核注册函数 --- catch、ignore、mask；总体而言，信息量较小

消息队列（？），以字节为序列的通信

管道，基于内存文件的通信，区分数据写端与数据读端

限制管道大小，一般而言读进程快于写进程，子进程会继承父进程的管道。

管道只能单向通信且是一次性操作

Ordinary pipe（anonymous pipes） 只能介导父子进程间的通信，往往单向

Named pipes 进程都可以访问，双向（要求非父子进程关系）

可用于父子进程间的通信

Independent process 不可被其他进程影响

Cooperating process 可以（被）其他进程影响 --- 需要IPC

线程thread

Lightweight process轻量级进程

拥有些许运行的资源如PC，寄存器，栈，有执行状态

TCB线程控制块

PC、寄存器、栈、线程ID、优先级都是私有的；data section、code section、heap、系

统资源是共享进程的

Traditional（heavyweight process）只有一个thread的任务

Dispatch 调度

进程是资源拥有单位，但不一定是调度单位

SMP机器，对称多处理结构机器，一个计算机上有一组处理器

用户级线程，不依赖于内核，应用进程自己维护；一个线程阻塞整个进程等待；

内核级线程，一个线程阻塞不影响其他线程；时间片分配给线程；

内核空间与用户空间线程对应方式

一对多，single kernel thread

优：不需要OS支持，个性化调度实现，调度开销较低

缺：不能使用多处理机，阻塞问题

一对一

优：并行运行，阻塞不会相互影响

劣：线程调度消耗较高，对于线程数量增多需要更好的扩展性

多对多，一般要求用户线程数量需要大于内核线程数量

Implicit thread 隐式多线程：将线程的创建与管理交给编译器和运行时库来完成

Fork() 执行同一程序的不同代码部分

Exec() 一个进程执行不同程序（往往接于fork()调用后使用）

Asynchronous 立即取消目标线程

Deferred cancellation 允许目标线程周期性判断自己是否应被取消

Thread pools：Create a number of threads in a pool where they await work

调用比创造省时；可以限制最大线程数

Process Schedule

Long term schedule 选择什么task放入main memory中

Short term schedule 选择什么task进行running

Preemptive 抢占式调度，可以任意调度

Non-preemptive 非抢占式调度，等到进程完成或发生阻塞才调度

Turnaround time（周转时间）：从进程提交到完成所经历的时间（CPU执行 + 等待时间等）

带权周转时间 = 周转时间 / CPU执行时间

Response time响应时间：从进程提交到首次被响应

Waiting time等待时间：进程在就绪队列中等待时间之和

Throughout 吞吐量：单位时间内完成的进程数

CPU utilization 处理机利用率

Gantt图（分块矩形）

FCFS 先来先服务调度

按task提交顺序进行调度，知道阻塞或运行结束才调度

利于长进程，利于CPU型进程

SJF 短作业优先调度

对当前预计执行时间最短的作业进行优先调度，有**最短的平均等待时间**

preemptive --- shortest remaining time first（SRTF）最短剩余时间优先（平均等待时间更短）

Highest response ratio next（HRRN）高响应比优先调度算法（不可抢占）

响应比R = （已经等待时间 + 要求执行时间） / 要求执行时间

Priority 优先权调度（有preemptive与否之分）

最高优先权进程优先调度，一般而言数值越小优先级越高

静态优先权，进程运行时不再改变

动态优先权，随时间改变而改变

使用aging（老化 --- 动态优先级）解决starvation问题

Round robin（RR）时间片轮转调度

Multilevel queue多级队列调度

根据性质/类型不同，将就绪队列分为若干子队列；不同队列对映不同的优先级、时间片长度、调度策略等（如foreground前台交互式处理和background后台批处理之分）

队列间需要有固定的优先级调度，并给定时间片长度

Multilevel feedback 多级反馈队列调度

多个就绪队列不同优先级（队列数值越小，优先级越高）；进程依次放入不同优先级队列，若队列1无法运行结束，则放置于队列2的结尾...... 每个优先级队列仅执行一个时钟周期（RR算法调度）

只有高优先级队列为空时，才调度低优先级队列；使用抢占式调度，被抢占的进程放入原队列的末尾

在I/O完成后，放回优先I/O请求时离开的队列，以免每次都回到最高优先级队列后再逐次下降（？）

Time slicing 时间片

Process synchronization

☆☆ 对映PPT P13例题 ☆☆

互斥mutual exclusion，多个进程不能同时使用同一资源

死锁 deadlock，永远得不到资源

饥饿 starvation，资源分配不公平

进程的最基本特性：并发性与独立性

互斥不能同时访问，同步需要联合使用

Critical section 临界区，访问临界资源的code

Entry section（进入区）

Critical section（临界区）

Exit section（退出区）

Reminder section（剩余区）

临界资源 一次只允许一个进程访问使用的词语

临界区并发解决方案一定要满足

Mutual exclusion

Progress空闲让进

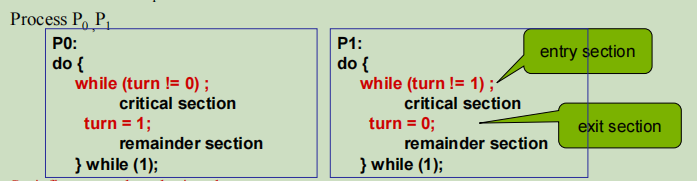
Bounded waiting 有限等待

让权等待（不是必须的）

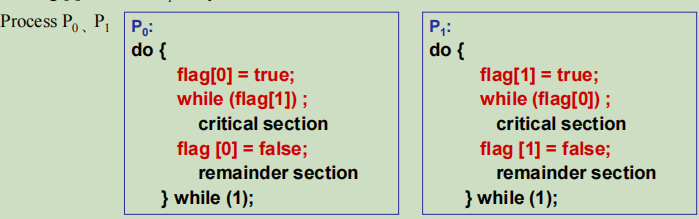
软件方法：Peterson’s solution

原子操作，load and store

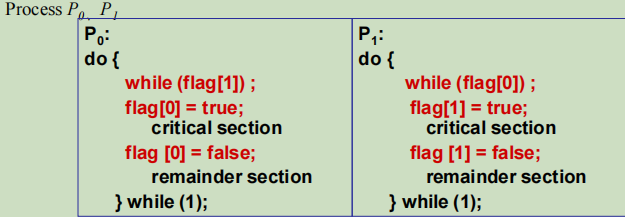
1.只有一个参数，强制轮流进入临界区

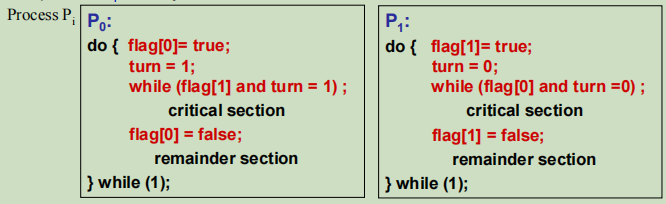


1. 两个参数 可能都无法进入



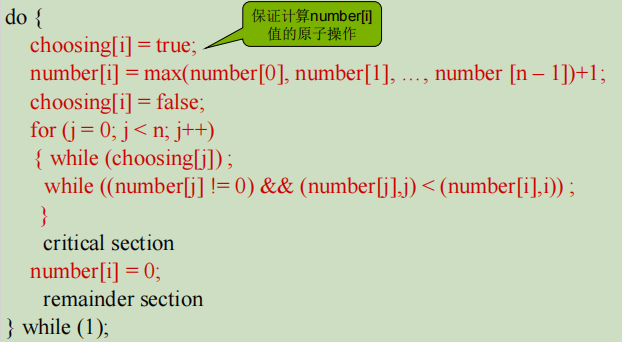
可能同时进入临界区



1. Peterson solution 2 + 1个参数（对于单核处理器才相对有效）

面包房算法（bakery algorithm）：process随机获得一个number，服务根据number大小（小优先）以及到达先后（先优先）进行。

Choosing用于控制是否处于number获取状态（这一状态不可被抢占）



硬件措施

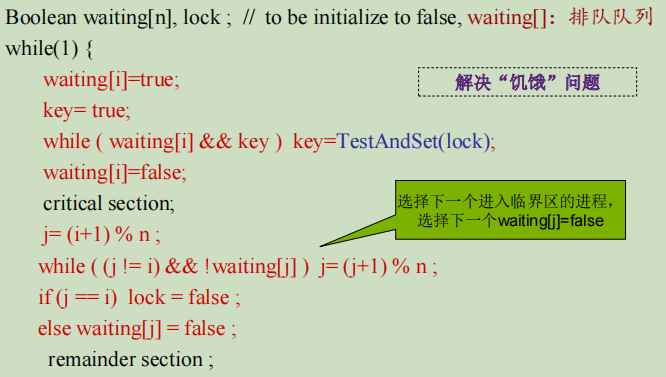
Disable interrupt，杜绝preemptive，原子性操作

TestAndSet（key），返回key的boolean值并将其置为true

Swap原子操作

优点：适用于任意数目的进程，简单易验证正确性；支持多个临界区，每个临界区设立一个布尔变量即可

缺点：不能实现让权等待，随机选择可能导致饥饿，可能导致死锁（优先级抢占式）



Spinlock 自旋锁，共用数据结构有关的锁定机制

不会被抢占，且需要被允许继续执行

Semaphores 信号量（不需要busy waiting）

Two standard operations（原子性操作） P()：wait()/down()作减法；V()：signal()/up()作加法；wait()操作相邻时，同步的wait()操作往往先于互斥的wait()操作

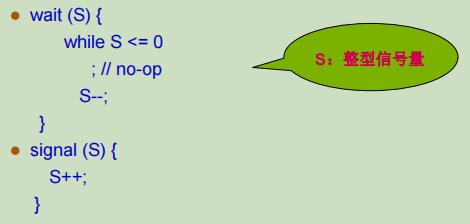
优点：简单

缺点：不安全，可能导致死锁，实现复杂

整型信号量（可能有busy waiting的弊端）

Counting semaphore计数信号量，可以range 一个domain

Binary semaphore 二值信号量，mutex locks

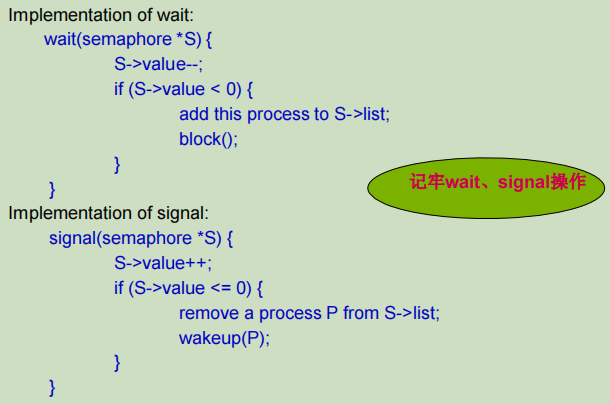


为每一个信号量增加waiting queue

增添两个操作（实际操作中符号注意）

Block() place a process on the appropriate waiting queue

Wakeup() remove a process from the waiting queue，place in the ready queue



记录性信号量

AND型信号量

Deadlock 死锁，注意：需要two or more processes 处于资源互相等待的阻塞状态

Deadlock：A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set.

Rampant 泛滥的，多发的

Monitor 管程是关于共享资源的数据结构及一组针对该资源的操作过程所构成的软件模块

需要为每个共享资源设立一个管程

信号量使用缺点

分散使用可能引起死锁

易读性差

不利于修改和维护

正确性难以保证

管程的主要特性

模块化，抽象数据类型（数据 + 操作），信息封装（半透明实现）

每次只能有一个进程进入管程，在管程中执行等待操作时，需要释放管程的互斥权

当管程中有两个进程时（一个从wakeup获得），一等待一运行（可自选等待进程）

Hoare 唤醒进程等待

Lampson redell 被唤醒进程等待

入口等待队列 entry queue

紧急等待队列：被唤醒进程的执行等待队列；优先级高于入口等待队列

条件变量：管程中阻塞进程原因

X.wait() 阻塞自身

X.signal() 唤醒自身

死锁产生的四个条件

Mutual exclusion（互斥）

Hold and wait（占有并等待，请求和保持）

No preemptive（不可抢占）

Circular wait（循环等待）

Resource allocation graph资源分配图

由process与resource（可以是一个集合 --- 对映多个同类资源）构成顶点集合

Request edge 请求边，由process指向resource（期望分配）

Assignment edge 分配边，由resource指向process（已分配）

If graph contains no cycles，there is no deadlock（无环无deadlock）

If graph contains a cycle（有环也不一定是deadlock）

if only one instance per resource type, then deadlock.

if several instances per resource type, possibility of deadlock.

死锁定理：S为死锁状态的充分条件是，尚且仅当S状态的资源分配图是不可完全简化的

死锁的处理策略

Prevention预防（打破死锁的必要条件）

Mutual exclusion -- 共享resource

Hold and wait 资源静态预分配方式，进程在运行之前获取所有需要的资源；或在仅有一个process时可持续获得资源

No Preemptive 转变为可抢占式

Circular wait资源的有序申请

Avoidance避免 声明最大resource使用量，不断检测剩余resource的使用状态

Safe state --- 安全序列（一般构建安全序列时优先满足序号小的process）若不存在安全序列 --> 不安全状态

If a system is in unsafe state，possibility of deadlock.（不安全不一定会死锁）

Avoidance ensure that a system will never enter an unsafe state.

Single instance -- resource allocation graph（资源分配图算法）

Multiple instance -- banker’s algorithm（银行家算法）

资源分配图算法

Claim edge（dashed line）需求边，可以转变为request edge和assignment edge

只有当需求边转变为分配边且图不会成环时，才允许申请

银行家算法

Available（当前可分配资源 --- work）、max（total need）、allocation（已分配资源）、need（仍需资源）矩阵

Safety algorithm：Process Request需小于need、available矩阵，预分配资源后判断是否处于safe state

Detection检测

Wait-for graph，process间的相互等待，查看是否成环

也是借助银行家算法判断安全性

☆Recovery解除

进程终止，终止所有进程 / 依次终止进程直至deadlock解锁

选择终止进程

Priority

Process已运行时长及完成仍需时间

Process资源使用 / 仍需请求

是否为交互式处理process

抢占资源

修复

内存管理，内存管理的核心是将logical address bind to separate physical address

Logical address 逻辑地址，相对地址，虚地址（virtual address）

Base register 基址寄存器（逻辑地址的起始）

Limit register 限长寄存器（单片逻辑地址的长度限制）

Physical address 物理地址，绝对地址，实地址

Address bind地址绑定，重定位 --- 将逻辑转换为物理地址

Compile time：绝对位置定位

Load time（静态重定位）：整块存入内存，相对起始地址编值（一旦装入后不可再改变位置）

Execution time（动态重定位）：需要硬件支持，程序运行时需要才重定位（可以逐指令）

Memory management unit（MMU）

介导虚拟地址到物理地址的映射 --- relocation register（重定位寄存器）

Dynamic loading，外部library只有需要时才装入memory中

Dynamic linking，library可以被预先放入（shared）memory中，只调用时物理地址才被计算；

Swapping（交换技术）中级调度

Backing store：large enough to accommodate copies of all memory images

Roll out换出：程序从内存移到辅存

Roll in换入：程序从辅存移到内存

一般的操作系统（linux，unix，windows）

正常情况下，禁止交换

 当空闲内存低于某一个阈值时，启用交换换出

 当空闲内存增加一定数量时，停止换出

Contiguous allocation 连续分配 into two partitions（只有一个用户进程）

Resident operating system

User process（含有base register，limit register，MMU）

Multiple partition allocation多分区分配

Fixed partitioning（固定分区）大小可以相同，可以不同 ---> 内部碎片

Dynamic partition（动态分区）--- variable partition（可变分区）

切出大小合适的区域分配给内存 --- 外部碎片

Allocated partitions

Free partitions

对映三种算法：（前两种的运行速度与存储利用率比第三种好）

First-fit：第一个足够大的

Best-fit：剩余最小的，search list

Worst-fit：剩余最大的，search list

Next-fit：在first-fit的基础上，并不每次都从第一个开始查找

Internal/external fragmentation 内部/外部碎片

Compaction/defragmentation 紧缩，拼接

Paging（分页管理）

Frame - physical memory（fixed-sized blocks）

Pages - logical memory

Page table，将逻辑地址转变为物理地址（内部碎片）--- 一直存放于memory中

Logical address = page number + page offset

Page-table base register（PTBR）

Page-table length register（PRLR）

Translation look-aside buffer（TLB） --- cache（联想寄存器、快表）

Effective access time（EAT 有效访问时间）：TLB总要访问一次

若miss会有两次memory access，hit也有一次memory access

☆☆Shared code must appear in same location in the logical address space of all processes

Hierarchical page 分级页表（打破页表需要连续存储的限制）

2nd outer page table/Outer page table/inner page table

Hash page table 哈希页表

Virtual page hash to a chain of elements（具体结构如page table的entry）

Inverted page table 反向页表

将每一个real page作为index，virtual page作为entry元素

可以减少memory存储空间，但会增加search time

Segmentation（分段式管理）长度可以不一致

Logical address = segment number + offset（offset用于判断在段中的具体偏移位置）

Segment table（段表）也需要base register与limit register

Segment table base register（STBR）

Segment table length register（STLR）

段表包括：段号，段长，段基址（需要判断offset在段长之内）

段页式管理：程序先分成若干程序段，每一段分成若干等大的固定页

地址变换，至少需要访存3次

Logical address = segment + page number + offset

Virtual memory：Only part of a running program needs to be loaded into memory for execution

局部性原理：较短时期中instruction与data分别局限于一定区域中。

空间局部性

时间局部性

Demand paging（请求页式管理）

Lazy swapper – never swaps a page into memory unless page will be needed

Valid - invalid bit（i --> page fault）

页表项：物理块号，状态位P，访问字段A（一段时间内被访问的次数），修改位R/W，外存地址（在disk上的位置）

Page fault：重新执行当前触发指令

Page fault rate与程序运行效率无关

Service the page-fault interrupt（缺页中断服务时间）

Read in the page（缺页读入时间）

Restart the process（重启进程时间）

Copy on write（对于share数据而言，只有要修改时才复制）

Demand segmentation（请求段式管理）

Page replacement 页面置换（replacement policy）

FIFO page replacement：可能产生belady’s anomaly（more frames more page faults）

OPT 向后最远使用页面替换

LRU最近最久使用（最近最久没有使用的page）

硬件支持：

计数器：直接记录最后一次使用时间

栈：栈底被替换，每次使用放置于栈顶

移位寄存器：使用则最高位置1，定时右移，高位补0，数值最小可被替换

NRU（类似于LRU）

使用reference bit（引用位，访问位），当引用位为1时可被替换

Clock算法（二次机会（second chance）算法）

检查引用位，为0可以替换；为1则清零引用位；循环check所有page

FIFO顺序排序，选取后point移至下一项

改进clock算法（增强......）

使用引用位与修改位（引用/修改过则置1）

替换顺序为 00 -> 01 -> 10 -> 11（其他类似于clock算法，含清零操作）

LFU最不经常使用（被替换）算法 --- 次数

MFU经常使用（被替换）算法 --- 此次数

页面缓冲算法

FIFO算法选择被置换页，若未修改--->空闲页面链表；若被修改--->已修改页面链表

已修改页面可以聚集写回disk；空闲页面，已修改页面可以重新被读入memory中

需要读入新的页面时，直接选取空闲链表中首个页面（写入disk数据）即可

Allocation of frame

Fixed allocation

Equal allocation 平均分配算法

Proportional allocation 按比例分配算法（可以根据process的size分配）

Priority allocation

组合策略：

固定分配局部置换策略（测试）

可变分配全局置换策略

可变分配局部置换

Encompass 包含

Thrashing 抖动：page swapping过于频繁

Does not have enough pages,page-fault rate is very high.

解决：

改变局部置换算法限制抖动

挂起若干进程

在cpu调度时引入工作集算法

工作集算法：动态页面算法

工作集窗口：从指定时刻开始向前看几个memory reference

工作集（WS）：在工作集窗口中不同page的数量/标识（页集合）

WSS 页集合的页个数

L = S准则，产生缺页平均时间（L） = 系统处理进程缺页的平均时间（S）

Memory mapped files

提前把disk中的file读至memory中（按page大小） ---> 后续进程的I/O file操作相当于对memory进行访问；

不同process可以把相同file的相同page映射到相同的memory位置

Memory allocate：buddy system（连续physical page形成的fixed size segment）

Memory allocated using power-of-2 allocator

分配满足request的2次幂内存空间

Request以2次幂向上取整

若request小于available，split it into two next-level power of 2

Kernel structure

Slab is one or more physically contiguous pages

Cache consists of one or more slabs

Prepaging 预调页（预约式页面调度）

在进程开始运行时，将所需要的页一起调入内存中

TLB范围：通过TLB可访问的内存量

TLB search = TLB Size（个数的size） \* Page Size

I/O interlock：Pages must sometimes be locked into memory

文件的概念：存储在某种介质上的具有文件名的一组相关数据的集合

File attributes（文件属性） --- file metadata（文件元数据）

Name、type、location、size、protection、time等

Identifier，标识file system中的file

File operation

Create、write、read、open、close

Reposition（重定位/文件定位），文件位置指针的重新定位，不涉及文件的读/写 --- 对映单个文件内指向指针而言（file在内存中的重定位？）

Delete 删除文件，释放文件所占的存储空间并删除目录条目

Truncate 截断文件，文件所有属性不变，但删除文件内容，长度置0并释放空间（目录项依旧保持）

Open：move the content of file attribute to memory

File pointer to last operation location

File-open count：counter number of file is open

Disk location of the file

Access rights

Close：write from memory to directory structure in disk

File structure（文件逻辑/内部结构）

流文件结构（没有文件结构）

记录文件结构

Simple record structure

Complex structure

File Access method

Sequential access（顺序存取）

Direct access（直接存取）

Indexed sequential-access（索引顺序）

Directory structure（文件目录本身也是文件）

目录项，存放FCB（file control block）

Entity containing file system known as a volume

Disk can be subdivided into partitions（分区），一个partition对映一个文件系统，一块disk可以对映多个文件系统，一个文件系统可以使用多块disk

Organize directory

Efficiency

Naming

不同user可以use the same name for different files

Same file 可以拥有different names

Grouping

Single-level directory

一个directory for all user

难以进行file分类以及file的同名使用

Tow-level directory

Separate each directory for each user

不同用户可以使用相同的file name

还是没有grouping的能力

Tree-structure directories（树型目录）

Absolute path name 绝对路径

Relative path name 相对路径

Acyclic-graph directories（无环图结构目录）

硬链接（使用的继承）/软链接（借用使用）

可以有共享的文件

使用back pointer（逆向指针）来防止dangling pointer（悬挂指针）存在的可能

General graph directory（普通图结构目录）

只允许连接至file而不允许连接至subdirectories ---> 防止环的产生

使用Cycle detection algorithm

访问文件系统中的文件前，必须进行挂载操作。

挂载是将文件系统连接到文件系统层次结构中的特定位置（挂载点）的过程

挂载点的挂载会覆盖原有内容（基于子树的全部覆盖）

File sharing

基于同一个user的共享

Unix使用符号连接（软链接）/硬链接

Windows使用快捷方式

Network file system（NFS）基于network的file sharing

Protection

Three class：owner access，group access，public access

File system structure（reside on secondary storage -- disk）

 文件系统是操作系统中以文件方式管理计算机软件资源的软件、被管理的

**文件**和**数据结构**（如目录和索引表等）的集合

 文件系统是对一个存储设备上的数据和元数据进行组织的机制

分层结构：

逻辑文件系统（内存中）

所有结构数据

管理目录结构

文件控制块维护文件结构

文件组织模块（外存中）

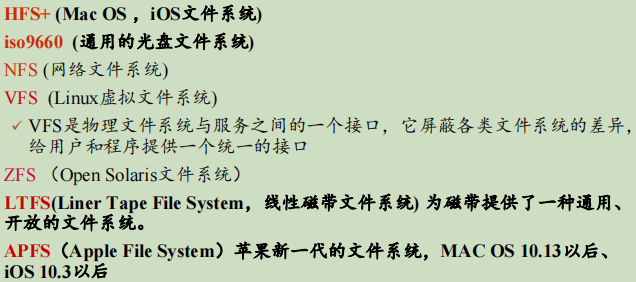
文件的逻辑块与物理快对映

基本文件系统

I/O控制（由设备驱动程序与中断处理程序组成）

一些file system types

NTFS为windows的file system



File system implementation

（On-disk）总块数、空闲块的数目和位置、目录结构、如何启动操作系统

File control block（FCB）文件控制块

Store the attributes about the file

File permission

File data

File owner，group，ACL（Access Control List）访问控制列表

File size

File data block（pointer）

（In-memory）

目录结构（directory structure）

分区表（partition table）

虚拟文件系统（virtual file system，VFS）

VFS允许在不同类型的文件系统上采用同样的系统调用接口（API）

API是针对VFS的接口，而非对任何特定类型的文件系统

Directory implementation

Linear list（线性检索法）线性搜索，耗时长

Hash table（哈希表）可能存在collision情况（碰撞，冲突）

哈希表的最大困难是其通常固定的大小和哈希函数对大小的依赖性

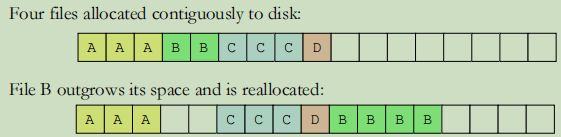
文件物理结构

Contiguous allocation（连续分配）

每个文件占据磁盘上的一组连续的块

记录起始位置 + 文件需要长度

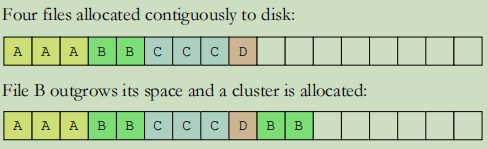
（文件大小难以动态增长）



Extent-based system（基于长度系统）

开始分配一块连续空间，当空间不够时，另一块被称为扩展的连续空间会添加到原来的分配中（离散化增加）

文件块的位置描述为开始地址、块数，加上指向下一扩展的指针。



Linked allocation（链接分配）

（隐式链接）

File为磁盘块的链表

可以使用簇（部分块的集合，只能成簇使用，更节省指针的占用空间）

（不能随机访问，块与块之间指针占用空间）

（显式链接）

FAT（file-allocation table）文件系统

根目录是一个普通的簇链，其目录项可以放在文件区任何地方

文件分配表，由“盘块号 + 下一块”组成，若无下一块则以 -1 填充

文件目录记录file的起始块号

Indexed allocation（索引分配）

将所有的数据块指针集中到索引块中

UNIX/Linux直接间接混合分配方法

前半部分为direct blocks

后半部分含有single indirect（一级），double indirect（二级），triple indirect（三级）等

FAT32磁盘结构

主引导记录MBR为主引导区的第一个扇区

主引导代码，分区表（最多4个条目）

计算机系统启动时，首先执行的是BIOS引导程序，完成自检，并加载主引导代码和分区表，然后执行主引导程序，由它加载激活分区（安装操作系统的分区）引导记录，再执行分区引导记录程序，加载操作系统，最后执行操作系统，配置系统。

NTFS file system

Free-space management空闲空间管理

空闲磁盘空间的记录

Bit map（易得连续文件）

链表，指向第一空闲块

分组：将n个空闲块的地址存在第一个空闲块中，而最后一块包含另外n个空闲块的地址，如此继续

计数：记录第一块的地址和紧跟第一块的连续的空闲块的数量n

Consistency checking

Log structure and transaction（事务）

低级格式化（物理格式化）

为每个扇区使用特殊的数据结构记录，包括校验码

分区

主引导记录分区表记录每个分区的起始扇区和大小（将磁盘分为由一个或多个柱面组成的分区，每个分区可以作为一个独立的磁盘）

逻辑格式化（创建文件系统）

一般顺序：磁盘的物理格式化 --- 对磁盘进行分区 --- 磁盘的逻辑格式化 --- 操作系统的安装（建立中断向量表）

Main memory以下的都属于大容量存储系统

Nonvolatile memory

Hard-disk memory

Magnetic tape

Disk

传输时间transfer time

Positioning time

寻道时间seek time

旋转延迟rotational latency

一般认为 寻道时间 ≈ 寻道距离

扇区是最小的I/O单元，扇区在磁道上长度可能不一致但存储大小一致，扇区与扇区之间往往有间隙（最外侧柱面的第一个磁道的第一个扇区 --- 0扇区）

磁道同心圆，由外向内从0开始

柱面：不同盘面相同半径磁道组成的集合

Disk attach（磁盘连接）

1. 通过I/O port连接
2. 通过network进行连接

DAS（direct（host）attached storage）直接连接存储

存储设备通过SCSI接口或光纤通道直接连接到一台计算机上（存储容量较为固定），借助与I/O port

NAS 网络连接存储：将存储设备通过标准的网络拓扑结构，连接到一群计算机上

通过LAN/WAN直接与NAS相连

SAN 存储区域网络：通过光纤通道连接到一群计算机上

先连接server，再通过SAN网络去storage array获得数据

伴有data-processing center / web content provider处理存取实际数据

Disk scheduling

FCFS 先来先服务，根据request序列进行偏移

SSTF 最短寻道时间，一组request集合中谁最近先服务谁（可能引起饥饿）

SCAN扫描算法（电梯算法），端到端移动，来回沿途均产生响应

LOOK算法，循环不回到0，仅回到满足首次需求的位置

C-SCAN循环扫描算法，端到端的单向移动，仅一次移动时沿途响应

C-LOOK算法，改进同LOOK算法

在heavy load时，SCAN与C-SCAN算法往往表现更好

SSTF和LOOK算法时默认的合理选择

Disk management

低级格式化（物理格式化）：将磁盘划分扇区

分区：把磁盘分成多个柱面组

逻辑格式化：创建文件系统

Swap-space management

Swap-space — Virtual memory uses disk space as an extension of main memory

Windows（normal file system） --- pagefile.sys文件

Linux （separate disk partition）-- 磁盘上专门的分区（SWAP分区）

RAID（redundant arrays of inexpensive（independent）disks）结构

更高的存储可靠性

更高的存取性能

Disk striping（条带化）use a group of disks as one storage unit

Mirroring（镜像化）

冗余磁盘备份数据

固态硬盘（SSD，solid state drives）

电子器件，存取速度较快，低功耗 / 非易失

Magnetic tape（磁带）

可靠性、数据传输速率超过磁盘

LTFS（linear tape file system）通用开放的磁带文件系统

I/O system

Computer的 two main jobs：I/O输入输出，processing

设备驱动程序为I/O子系统提供了统一接口

Devices 寻址方式

Direct I/O instruction 放置于memory之外的I/O专属区域

Memory-mapped I/O 放置于memory中

I/O端口：设备控制器（连接CPU与设备）

I/O接口：设备控制器中可被CPU直接访问的寄存器

数据输入/输出寄存器、状态寄存器、控制寄存器

I/O方式

程序I/O：polling轮询（字符传递）

中断I/O（同步/异步I/O）：（字符传递）

IRL（interrupt request line 中断请求线）

非屏蔽中断：处理不可恢复的内存错误事件

可屏蔽中断：由CPU在执行关键的不可中断的指令序列前加以屏蔽

中断优先级

DMA方式

数据块传递

不经过CPU进行device到memory的data传递

层次结构

用户级软件：统一的命令

与设备无关的操作系统软件：实现逻辑设备到物理设备的转变

设备驱动程序：原则上同类设备只需要一个，将来自上层软件的与设备无关的的抽象请求转为具体请求

中断处理程序

阻塞I/O ：等待数据准备与传递

非阻塞I/O：不断轮询，直至数据准备就绪后才传输

同步异步表现在主动去查看结束还是被提醒结束

用户程序设备独立性：不直接使用物理设备名，使用逻辑设备名

I/O软件设备独立性：软件不依赖于硬件

Buffer

解决设备速度不匹配

解决设备传输块大小不匹配

假脱机技术spooling 保存设备输出的缓冲

通过虚拟技术，将独享设备表现为可供多个用户进行共享的设备一样