操作系统概述

2022年6月28日 16:17

操作系统需要了解的内容包括：

1、操作系统概述
2、进程管理
3、内存管理
4、文件管理
5、输入输出管理

操作系统概述：

- 操作系统概念
  - 操作系统（OS）是管理计算机硬件和软件的一个程序，本身就是一个系统软件，是管理者的角色
  - 常见操作系统：Windows，DOS，unix（安卓和ios均基于unix）
  - 计算机系统的构成：用户>应用程序>操作系统>裸机
  - 操作系统向用户提供了一些访问方式，称为接口
  - GUI：图形用户接口
- 操作系统的功能
  - 系统资源的管理者
    - 处理器管理
    - 存储器管理
    - IO管理
    - 文件管理
  - 用户、软件与系统硬件之间的接口
    - 程序接口
    - 命令接口：一切在电脑上的操作，底层都是命令，如双击文件夹，即是调用dir命令，显示文件夹文件
    - GUI（图形用户接口），跟命令接口相比，更加直观
  - 实现了对计算机资源的抽象
    - 将具体的计算机硬件资源抽象成软件资源，方便用户使用
    - 开放了简单访问方式，隐藏了实现细节
- 操作系统的目标
  - 有效性，即提高效率
    - 提高系统资源的利用率
    - 提高系统的吞吐率
  - 方便些，方便用户使用
  - 可扩展性
  - 开放性
- 操作系统的特征
  - 并发
    - 同一时间间隔（一个时间段）内执行和调度多个程序的能力
      - 宏观上，一个处理器同时执行多个任务
      - 微观上，处理器在多个程序之间快速切换
    - 是虚拟和异步的前提，与共享互为前提

与此类似的一个概念：并行

同一时刻所发生的事件数量

有物理极限，因此更关注操作系统并发量

共享

即资源共享，系统中的硬件资源供多个并发执行的软件共同使用

方式：

同时访问方式：同一时间段允许多个软件同时访问共享资源，如硬盘

互斥共享方式：独占式，一个软件使用需要另一个软件先用完

虚拟

使用某种技术把一个物理上的实体变成多个逻辑上的对应物

时分复用技术TDM：将时间进行分割（四核八线程），分时交替进行

空分复用技术SDM：将空间进行分割（如一块磁盘分割出多个卷）

异步

程序执行的异步性，程序执行不是一贯到底的，而是走走停停

程序执行的不可预知性

多道程序环境下，允许多个程序并发执行

单处理器下，多个程序分时交替进行

操作系统的演变

分时操作系统：即分割时间片，为多个用户提供服务，本质上还是并发或者分时交替执行，缺点是没有任务优先级

实时操作系统：能即时响应外部请求并进行处理

微机操作系统：即个人pc的操作系统

网络操作系统

分布式操作系统：一个任务比较复杂，将其分给多个计算机共同完成，每个计算机之间没有等级关系，并行性

不同操作系统对内核的定义有些许不同，因此有微内核、宏内核等区分

内核是操作系统最基本的部分。它是为众多应用程序提供对计算机硬件的安全访问的一部分软件，这种访问是有限的，并且内核决定一个程序在什么时候对某部分硬件操作多长时间。内核的分类可分为单内核和双内核以及微内核。严格地说，内核并不是计算机系统中必要的组成部分，因为早期计算机是没有操作系统的。

操作系统的运行机制

- 时钟管理
  - 计时：提供系统时间
  - 时钟中断：如进程切换
- 中断机制：提高多道程序环境下cpu的利用率
  - 中断不一定是坏事
  - 分类：
    - 外中断：中断信号来源于外部设备（被迫）
    - 内中断：中断信号来源于当前指令（自愿）
      - 陷阱/陷入：由应用程序主动引发
      - 故障：由错误条件引发，处理完后继续执行后续指令
      - 终止：由致命错误引发
  - 中断处理机制
    - 产生故障
    - 关中断：到下一个开中断之前cpu不处理更高级别的中断请求，因为要保存现场
    - 保存断点：将原来的执行到哪了的地址给保存起来，用cpu里控制器里的PC（程序计数器）保存
    - 引出中断服务程序
    - 保存现场和屏蔽字
    - 开中断
    - 执行中断服务程序
    - 关中断
    - 恢复屏蔽和屏蔽字
    - 开中断
    - 中断返回
- 原语：是一个被封装起来的常用程序段
  - 由若干条指令组成，用来完成某一特定功能，执行过程不会被中断（原子性）
  - 原语运行在内核空间中
  - 不能中断的底层逻辑就是用开中断和关中断来实现的
- 系统数据结构
  - 进程管理
  - 存储器管理
  - 设备管理
- 系统调用
  - 由操作系统实现，给应用程序调用
  - 是一套接口的集合
  - 应用程序访问内核服务的方式

操作系统的体系结构

- 传统的操作系统结构
  - 第一代：无结构OS

第二代：模块化os

即将某些函数进行分类

模块独立性的标准：高内聚（某个模块所有操作均相似）、低耦合（模块之间联系不大）

第三代：分层式os

铺设若干个层次，每个层次职责不同，自底向上，更加可靠，但因为层次通信导致效率低

第四代：微内核OS

宏内核其实就是一个完整的操作系统

微内核：足够小的内核，只实现os的核心内容，只包含和硬件有关的东西，不是完整的操作系统

采用面向对象技术

应用“机制与策略分离”原理

机制：真正干活的

策略：管理者

优点：

提高os的可拓展性、可靠性、可移植性

支持分布式操作系统（宏内核也能，但成本高）

融入了面向对象技术

缺点：

效率不及早期宏内核

（用户态与核心态频繁切换，因此需要频繁地中断，即需要保存现场...等中断机制）

进程管理

2022年6月30日 16:53

进程

Process，是系统一个具有一定独立功能的程序关于某个数据集合的一次运行活动，是系统进行资源分配的基本单位或最小单位。

进程在内存中的结构

控制块（PCB），进程的唯一表示，相当于id，系统根据pcb来识别进程。无论是不是同一个程序产生的进程，pcb都是不一样的

数据段，存放原始数据和中间数据

程序段，存放在文本区域，可被多个进程共享。同一个应用程序的多个进程共享

进程的特征

动态性：由创建而生，由撤销而亡

并发性：多个进程同时运行

独立性：独立资源分配

异步性：相互独立、互不干扰（但可以进行数据的交互）

线程：Thread，进程的轻型实体，也叫轻量级进程，是一系列活动按实现设定好的顺序依次执行的活动，是一系列指令的集合，不能单独存在，必须包含在进程程中，是OS运算调度最小单位。

注意：线程目前可以继续划分，成纤程

引入线程的原因：提高OS的并发性，并减少资源的分配

线程相较于进程，大大降低了创建、撤销和切换可执行实体的成本和难度

线程的实现方式

用户级线程：线程在用户空间中实现，线程控制块在用户空间

内核级线程：线程在内核空间中实现，线程控制块在内核空间

句柄（handle或HWND）：指针的指针，Windows通过进程PID找到进程句柄，通过句柄定位进程内存地址

Windows是一个以虚拟内存为基础的操作系统，很多时候，进程的代码和数据并不全部装入内存，进程的某一段装入内存后，还可能被换出到外存，当再次需要时，再装入内存。两次装入的地址绝大多数情况下是不一样的。也就是说，同一对象在内存中的地址会变化。那么，程序怎么才能准确地访问到对象呢？为了解决这个问题，Windows引入了句柄。

系统为每个进程在内存中分配一定的区域，用来存放各个句柄，即一个个32位无符号整型值（32位操作系统中）。每个32位无符号整型值相当于一个指针，指向内存中的另一个区域（我们不妨称之为区域A）。而区域A中存放的正是对象在内存中的地址。当对象在内存中的位置发生变化时，区域A的值被更新，变为当前时刻对象在内存中的地址，而在这个过程中，区域A的位置以及对应句柄的值是不发生变化的。这种机制，用一种形象的说法可以表述为：有一个固定的地址（句柄），指向一个固定的位置（区域A），而区域A中的值可以动态地变化，它时刻记录着当前时刻对象在内存中的地址。这样，无论对象的位置在内存中如何变化，只要我们掌握了句柄的值，就可以找到区域A，进而找到该对象。而句柄的值在程序本次运行期间是绝对不变的，我们（即系

统）当然可以掌握它。这就是以不变应万变，按图索骥，顺藤摸瓜。

1.所谓“唯一”、“不变”是指在程序的一次运行中。如果本次运行完，关闭程序，再次启动程序运行，那么这次运行中，同一对象的句柄的值和上次运行时比较，一般是不一样的。

其实这理解起来也很自然，所谓“一把归一把，这把是这把，那把是那把，两者不相干”（“把”是形象的说法，就像打牌一样，这里指程序的一次运行）。

2.句柄是对象生成时系统指定的，属性是只读的，程序员不能修改句柄。

3.不同的系统中，句柄的大小（字节数）是不同的，可以使用sizeof()来计算句柄的大小。

4.通过句柄，程序员只能调用系统提供的服务（即API调用），不能像使用指针那样，做其它的事

进程的运行

三种基本状态

就绪（ready）：可运行但还未运行的状态，获得了除cpu以外的所有条件

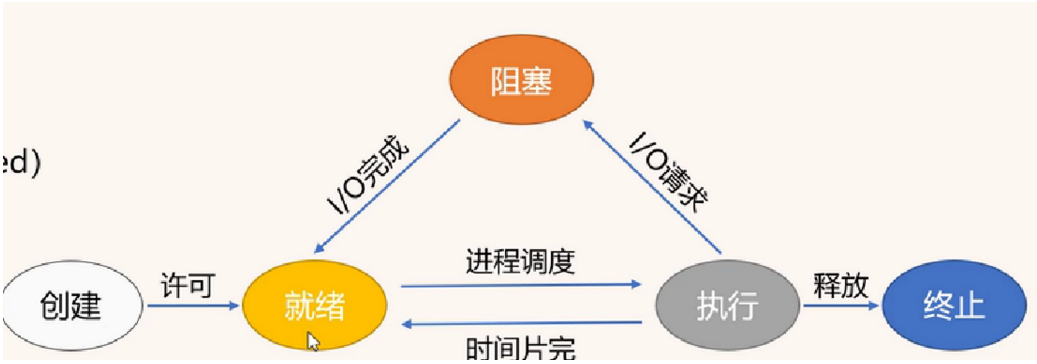
执行（running）：就绪+获得cpu使用权

阻塞（blocked）：执行之际突然需要IO请求等原因

另外两种状态：

创建（new）

终止（terminated）



进程控制

即OS对进程实现有效的管理，包括创建新进程、撤销已有进程、挂起等多种操作，OS通过原语（Primitive）操作实现进程控制

原语：是一种原子操作，由若干条指令组成，完成特定的功能

原语的特点：

要么全做、要么全不做，执行过程中不会被中断

在管态/系统态/内核态中执行，常驻内存

是内核三大支撑功能（中断处理/时钟管理/原语操作）之一

挂起与激活

suspend挂起：当前的进程不再运行了，便于用户分析进程等

如锁屏程序，一般都是挂起的，只有锁屏时才会运行

进程执行期间不能被直接挂起，必须转换成就绪或者阻塞

active激活

进程的调度

处理器（处理机）调度

根据一定的算法和原则将处理器资源进行重新分配的过程

前提：作业/进程数远远大于处理机数

目的：提高资源（主要是cpu）利用率，减少处理机空闲时间

调度程序：一方面要满足特定系统用户的需求（如快速响应），另一方面需要考虑系统整体效率和调度算法本身的开销

调度的层次

高级调度/作业调度

把后备作业调入内存

只调入一次，调出一次

中级调度/内存调度

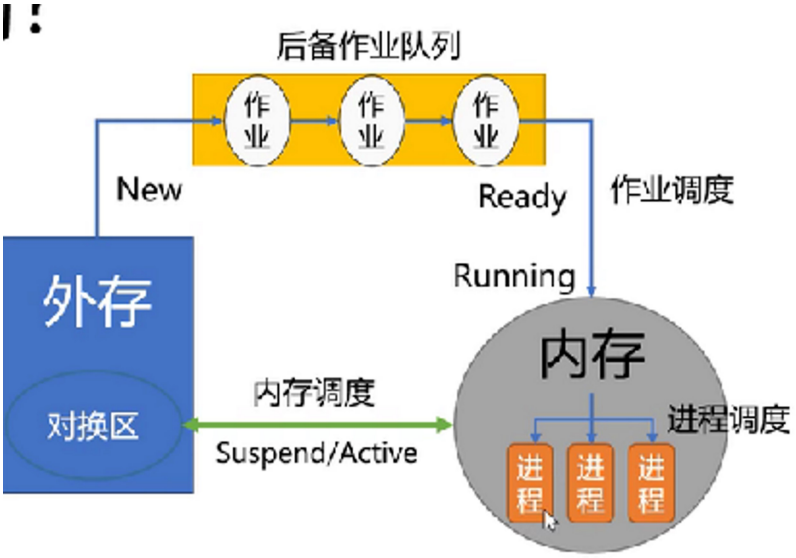
将进程调至外存，条件合适再调入内存

在内、外存对换区进行进程对换

低级调度/进程调度

从就绪队列选取进程分配给处理器

最基本的调度，频率非常高（相当于一个时间片完成）



剥夺式/抢占式调度

立即暂停当前进程

分配处理机给另一个进程

原则：优先权/短进程优先/时间片原则

非剥夺/非抢占式调度

等待，直到当前进程完成或者阻塞

特点：适用于批处理系统，不适用于分时、实时操作系统

调度的时机

进程运行完毕

进程时间片用完

进程要求IO操作

执行某种原语操作

高优先级进程申请运行

调度的过程

保存镜像：记录进程现场过程

调度算法：确定分配处理机的原则

进程切换：分配处理机给其他进程

处理机回收：从进程收回处理机

调度算法指标

cpu利用率

系统吞吐量

周转时间

等待时间

响应时间

调度算法：

先来先服务（FCFS）

调度作业/就绪队列中最先入队者，等待操作完成或阻塞

原则：按作业/进程到达顺序服务（执行）

非抢占式

优缺点：

有利于cpu繁忙型作业，充分利用cpu资源

不利于IO繁忙型作业，操作耗时，其他饥饿（得不到执行）

短作业优先（SJF）

所需服务时间最短的作业/进程优先服务

原则：追求最少的平均（加权）周转时间

非抢占式

优缺点：

平均等待时间、周转时间最少

长作业周转时间会增加

估计时间不准，不能保证紧迫任务及时处理

高响应比优先调度（HRRN）

结合短作业优先和先来先服务，综合考虑等待时间和运行时间计算响应比，高的优先调度

原则：综合考虑作业/进程的等待时间和服务时间

非抢占式

响应比：（等待时间+服务时间）/服务时间

优缺点：

长作业等待时间越久，响应比越高

当一个程序执行完成或阻塞时，均需重新计算每个程序响应比

优先级调度（PSA）

按作业的紧迫性进行调度

原则：优先级最高的作业先调度

抢占式或非抢占式

优先级设置原则：

静态/动态优先级

系统>用户（系统要提供服务）；交互>非交互；IO型>计算型

缺点：优先级低的可能永远得不到执行

时间片轮转调度（RR）

按进程到达就绪队列的顺序，轮流分配一个时间片取执行，时间片用完则剥夺

原则：公平、轮流为每个进程服务，进程在一定时间内都能得到响应

抢占式，由时钟中断确定时间

时钟中断：进程运行一段时间片的时间，时间一到，硬件就会产生一个时钟中断，cpu执行权就切换了

优缺点：

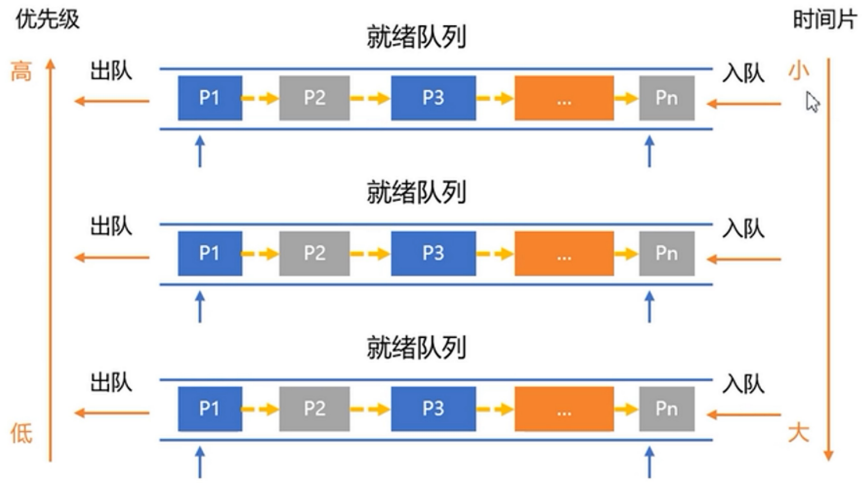
公平、响应块，适用于分时系统（注意分时和实时的区别）

时长的决定：系统响应时间、就绪队列进程数量、系统处理能力

时间片太大，就相当于FCFS，太小则处理过于频繁，开销增大

多级反馈队列调度（MFQ）

设置多个按优先级排序的就绪队列



原则：集前几种算法的优点于一身，相当于SPA+RR

抢占式

优缺点：

对各类型相对公平，快速响应

终端型作业用户：就会短作业优先

批处理作业用户：就会周转时间短

长批处理作业用户：在前几个队列部分执行

进程之间的协作

进程通信：进程之间的信息交换

进程是资源分配的最小单位，各进程内存空间彼此独立

一个进程不能随意访问其他进程的地址空间

三种方式（一台计算机内部）

共享存储（shared-memory）

比较早的实现方式，会有安全问题

基于共享数据结构的通信方式

多个进程共用某个数据结构（OS提供并控制）

由用户负责同步处理

低级通信：可以传递少量数据，效率低

基于共享存储区的通信方式

多个进程共用内存中的一块存储区域

由进程控制数据的形式和方式

高级通信：可以传递大量数据，效率高

消息传递

直接通信：点到点发送

间接通信：广播信箱（以信箱为媒介）

管道通信（pipe）

管道：用于连接读/写进程的共享文件

本质是内存中固定大小的缓冲区

高效、传输数据量大，基本无安全问题

半双工通信

特点：同一时段只能单向通信，即一条管道只能有一个方向

所以如果需要双向同时传输，则需要两条管道

以先进先出的方式组织数据传输

进程同步

协调进程间的相互制约关系，使它们按照预期的方式以及顺序去执行的过程

制约的形式

间接相互制约关系（互斥）：进程排他性地访问共享资源

互斥的访问临界资源（共享资源）

硬件实现

软件实现

直接相互制约关系（同步）：进程间的合作，比如管道通信

管程：或者叫监视器，是进程同步的一个工具，是一个管理者

是有低矮表共享资源的数据结构和一组方法（函数）组成的管理程序（封装）

管程可以理解为管理程序

组成：

管程名称

局部于管程内部的共享数据结构

对该数据结构操作的一组函数

管程内共享数据的初始化语句

死锁

多个进程由于竞争资源而造成的阻塞现象，若无外力作用，这些进程将无法继续推进，卡死

相似概念：饥饿：等待时间过长以至于给进程推进和响应带来明显影响

死锁产生的原因：

- 系统资源的竞争
- 进程推进的顺序非法

死锁预防（程序运行之前避免）：

- 破坏互斥条件：将互斥访问的资源改成允许同时访问（不是所有资源都能共享）
- 破坏不可抢占的条件：将某个进程占用的共享资源强制释放
- 破坏请求并保持条件：
  - 进程一开始一次性申请所有需要的资源
  - 阶段性请求和释放资源
- 破坏循环等待条件：按序号请求资源

死锁避免（程序跑起来后避免）：安全性算法

死锁的处理策略：

- 死锁的检测与解除
  - 死锁的检测
  - 死锁的解除
    - 资源剥夺
    - 撤销进程
    - 进程回退

内存管理

2022年7月3日 13:24

多道程序：多个程序同时进入内存开始运行，道可以简单理解为个
分区，可以简单理解为分块，也就是把内存划分为数个区域，每个区域只能装入一个进程

存储器的多层结构
- 寄存器
- 高速缓存
- 主存储器
- 硬盘缓存
- 固定磁盘
- 可移动存储介质

进程运行的基本原理
- 程序->进程：
  - 编译：将我们写的代码编译成计算机的二进制指令
    - 不属于操作系统的工作，是人来完成的
  - 链接：将我们写的代码模块和所需要的库整合、打包、封装
    - 结果是产生一个装入模块
    - 包括：
      - 静态链接：适用于小的程序，在装入之前就完全链接好了
      - 装入时动态链接：一边装入一边链接
      - 运行时动态链接：运行的时候，若出现缺页中断（所需要的内容不在内存中），此时再进行链接和装入
  - 装入：把装入模块加载到内存中
    - 三种分类：
      - 绝对装入：写程序时即固定了程序在内存中的位置，不再使用
      - 可重定位装入：根据内存，选有空闲的地方
      - 动态运行时装入：以上两种都是一次性全部将程序加载到内存中，不存在运行时还需要加载新的东西到内存
      - 而这种方法则是在运行时加载新东西到内存中，且一个程序可能分配到了不连续的内存空间中。（发现需要的东西内存中没有，要从磁盘新加载进来）
    - 两个细节：
      - 逻辑地址与物理地址
      - 内存保护
        - 装入当前程序的过程中保护此内存地址不受其他程序的影响，防止数据紊乱
    - 内存扩充技术：
      - 内存空间比程序所需要的空间小，这种技术可以实现小内存运行大程序
      - 两种方式：
        - 覆盖：
        - 覆盖的基本思想是：由于程序运行时并非任何时候都要访问程序及数据的各个部分（尤其是大程序），因此可以把用户空间分成一个固定区和若干个覆盖区。将经常活跃的部分放在固定区，其余部分按调用关系分段。首先将那些即将要访问的段放入覆盖区，其他段放在外存中，在需要调用前，系统再将其调入覆盖区，替换覆盖区中原有的段。
        - 覆盖技术的特点是打破了必须将一个进程的全部信息装入主存后才能运行的限制，但当同时运行程序的代码量大于主存时仍不能运行。
        - 交换：
        - 交换（对换）的基本思想是，把处于等待状态（或在CPU调度原则下被剥夺运行权利）的程序从内存移到辅存，把内存空间腾出来，这一过程又叫换出；把准备好竞争CPU运行的程序从辅存移到内存，这一过程又称为换入。中级调度就是采用交换技术。

内存的分配
- 连续分配方式：所分配的内存空间是连续的
  - 单一连续分配
    - 单一指单用户单进程
    - 优点：实现简单、无外部碎片、不一定需要内存保护
    - 缺点：只能用于单用户、单任务的操作系统，有内部碎片，存储器利用

率低

固定分区分配

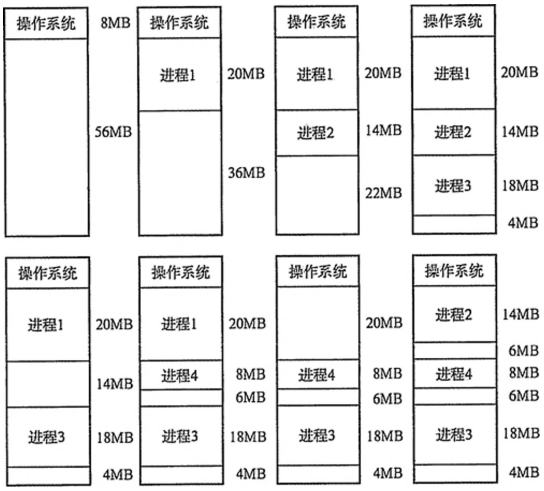
固定分区分配是最简单的一种多道程序存储管理方式，它将用户内存空间划分为若干个固定大小的区域，每个分区只装入一道作业。当有空闲分区时，便可以再从外存的后备作业队列中,选择适当大小的作业装入该分区，如此循环。注意固定的含义，固定分区之间的大小可以相等也可以不等

优点：实现简单、无外部碎片

缺点：较大用户程序时，装不下，需要采用覆盖技术，降低了性能、程序太小也要占用一个内存分区就会浪费空间，即产生产生内部碎片，空间利用率低

动态分区分配

动态分区分配又称为可变分区分配，是一种动态划分内存的分区方法。这种分区方法不预先将内存划分，而是在进程装入内存时，根据进程的大小动态地建立分区，并使分区的大小正好适合进程的需要。因此系统中分区的大小和数目是可变的。



如图，进程虽然一开始是紧跟着的，但是随着中间进程结束而上下进程没有结束时，便产生了割裂，并进而产生了碎片

缺点：

随着时间的推移，内存中会产生越来越多的碎片，内存的利用率随之下降。这些小的内存块称为外部碎片，指在所有分区外的存储空间会变成越来越多的碎片，这与固定分区中的内部碎片正好相对。克服外部碎片可以通过紧凑（Compaction)技术来解决，即操作系统不时地对进程进行移动和整理。但是这需要动态重定位寄存器的支持，且相对费时。

分配算法：

进程装入或换入主存中，若内存中存在多个足够大的空闲块，操作系统必须确定分配哪个内存块给进程使用，即分配策略

首次适应算法：

空闲分区以地址递增的次序链接。分配内存时顺序查找，找到大小能满足要求的第一个空闲分区。

最佳适应算法：

空闲分区按容量递增形成分区链，找到第一个容量能满足要求的空闲分区。

最坏适应算法：

又称最大适应(Largest Fit)算法，空闲分区以容量递减的次序链接。找到第一个能满足要求的空闲分区，也就是挑选出最大的分区。

领近适应算法：

又称循环首次适应算法，由首次适应算法演变而成。不同之处是分配内存时从上次查找结束的位置开始继续查找。

非连续分配方式

非连续分配允许一个程序分散地装入到不相邻的内存分区中，根据分区的大小是否固定分为分页存储管理方式和分段存储管理方式，根据运行作业时是否要把作业的所有页面都装入内存才能运行分为基本分页存储管理方式和请求分页存储管理方式。

基本分页存储管理方式

固定分区会产生内部碎片，动态分区会产生外部碎片，这两种技术对内存的利用率都比较低。我们希望内存的使用能尽量避免碎片的产生，这就引入了分页的思想：把主存空间划分为大小相等且固定的块，块相对较小，作为主存的基本单位。每个进程也以块为单位进行划分，进程在

执行时，以块为单位逐个申请主存中的块空间。尽管会产生内部碎片，但是这种碎片相对于进程来说也是很小的，每个进程平均只产生半个块大小的内部碎片（也称页内碎片）。

进程中的块称为页(Page)，内存中的块称为页框（Page Frame，或页帧）。外存也以同样的单位进行划分，直接称为块(Block)。进程在执行时需要申请主存空间，就是要为每个页面分配主存中的可用页框，这就产生了页和页框的一一对应。而页面则是页和页框的大小。页面应该是2的整数幂且大小适中。

地址结构：包含页号和页面偏移量

页表：为了在内存中找到进程每个页所对应的物理块，系统为每个进程建立一张页表，记录页在内存中对应的物理块号，页表一般存放在内存中。页表的作用时实现从页号道物理块号的地址映射。页表一般放在主存中。

快表：由于取数据或指令需要先访问页表，然后再根据页表地址来取实际的指令，这样速度较慢。又称联想寄存器TLB，是一个高速缓冲存储器，加速地址的变换。而存放在主存的页表也称慢表。

二级页表：用来解决页表占用存储过大的问题

基本分段存储管理方式

按用户进程中的自然段划分逻辑空间

例如，用户进程由主程序、两个子程序、栈和一段数据组成，于是可以把这个用户进程划分为5个段，每段从0 开始编址，并分配一段连续的地址空间（段内要求连续，段间不要求连续，因此整个作业的地址空间是二维的）。

地址结构：包含段号和段内偏移量

同样有段表

能够反映程序的逻辑结构并有利于共享

段页式管理方式

将分页和分段结合起来

将地址空间先分段再分页，地址结构为：段号、页号、页面偏移量

系统为每个进程建立一张段表，每个分段一张页表，也就是每个进程段表只有一张，但页表可能有多个

虚拟内存（基本东西跟上面是一样的）

具有请求调入和置换功能、从逻辑上对内存容量加以扩充的一种存储器系统

局部性原理：催生缓存技术

时间局部性：局部时间内多次操作同一个数据

空间局部性：多次访问某一个空间范围内的数据

虚拟内存的特征

多次性：多次调入内存（程序需要的部分就调入）

对换性（置换）：需要就调入，不需要就移出去

虚拟性：逻辑上的概念，是逻辑上的扩充，内存+外存即虚拟内存

虚拟内存的实现

请求分页存储管理

页表机制

缺页中断机制（要的东西在内存中没有，必须去外存加载即缺页）

地址变化机制

页面置换算法：要加载新东西到内存但内存又满了，所以需要置换

先进先出法

时钟置换算法

最佳置换算法：淘汰最不可能再次使用的页面（无法实现）

最久最近置换法

改进型时钟置换算法

驻留集：程序在主存中占用的空间大小，用页数来表示

页面分配策略

固定分配局部置换

可变分配全局置换

可变分配局部置换

调入页面的时机

预调入策略

请求调页策略

请求分段存储管理

请求段页式存储管理

逻辑地址：从程序角度看到的内存单元、存储单元、网络主机的地址，也叫相对地址

物理地址：每一个字节单元给一个唯一的存储器地址，从0开始编号依次+1，又称绝对地址

逻辑地址与物理地址的转换

透明：也就是黑盒，指计算机中客观存在、运行的实体但我们却不可看到的特性，比如C++的加法，会用就行，而无需了解其底层实现原理

文件管理

2022年7月5日 0:22

文件的概念

以计算机硬盘位为主的存储在计算机上的信息集合

目录其实也是文件

属性：描述文件状态的一组信息，比如名称、类型、大小等

基本操作：创建、读、写、寻址、删除等

文件的结构

逻辑结构

无结构文件（流式文件）

没有所谓的明确的结构，我们能看的就是单纯的字节

只能一个个字节挨着个去操作

有结构文件（记录式文件）

包括：顺序文件、索引文件、索引顺序文件、直接或散列文件

文件的目录结构：

文件控制块FCB

索引结点

目录结构

物理结构

文件系统的层次结构

访问文件是系统调用，因为这是操作系统内核的功能

文件系统的实现

目录实现

线性列表

哈希表

文件实现

文件的分配方式

连续分配

链接分配

索引分配

文件存储空间管理

空闲表法

空闲链表法

成组链表法

位示图法

输入输出管理

2022年7月5日 0:45

IO设备分类

- 按使用特性
  - 人机交互类外部设备
  - 存储设备
  - 网络通信设备
- 按传输速率：低、中、高速设备
- 按信息交换单位：块设备、字符设备

IO设备的构成

- 机械部件：用来执行具体的IO操作
- 电子部件：即IO控制器，设备控制器，是CPU与硬件设备之间的桥梁

IO控制器的主要作用

- 接受并识别CPU命令
- 向CPU报告设备状态
- 数据交换
- 地址识别

IO控制器的组成

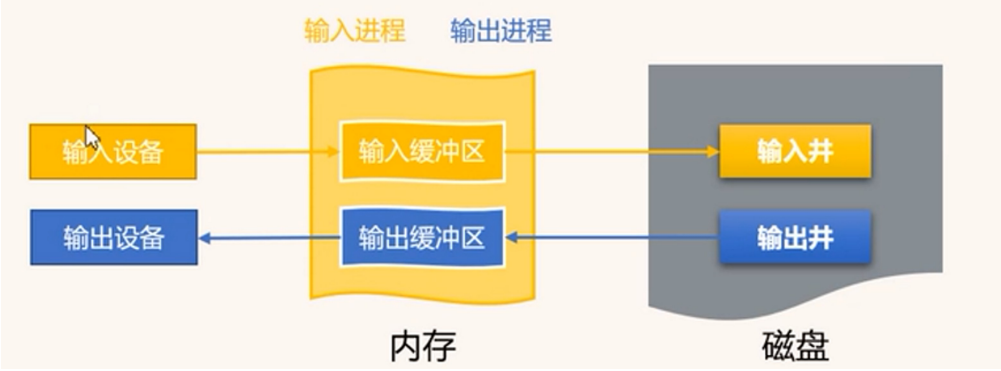
- CPU和控制器之间的接口
- IO逻辑
- 控制器与设备之间的接口

IO控制方式

- 程序直接控制方式
- DMA方式
- 通道控制方式

IO核心子系统

- IO调度
- 设备保护
- 假脱机技术(SPOOLing技术)



加入了缓存，人可以输入数据到缓冲区，然后缓冲区加载到磁盘，第二部看起来好像直接拷贝而没有人参加，所以是假脱机

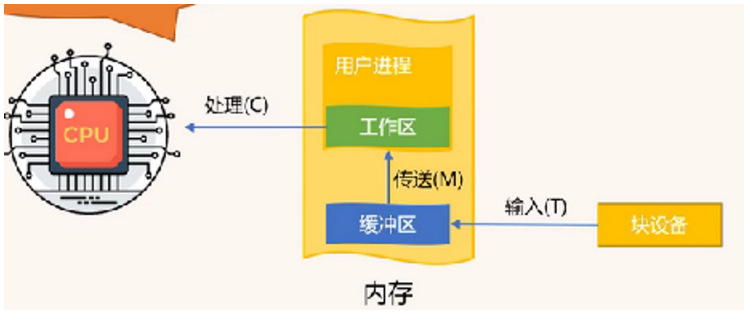
设备分配与回收

缓冲区管理（一般在内存中）

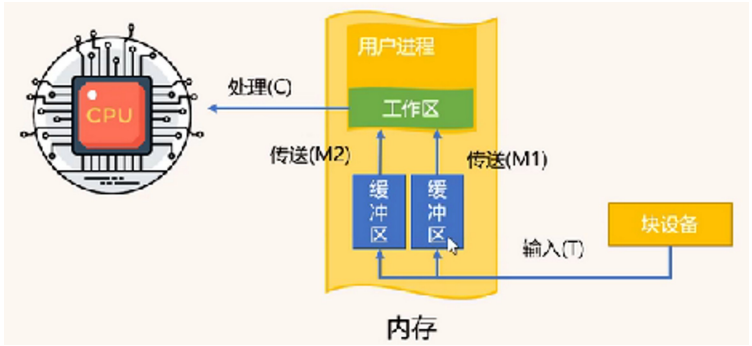
为缓解CPU和IO设备间速度不匹配的矛盾而建立的临时存储区域

分类（根据数量）：

单缓冲：非空不写、未满不读，也就是等一次冲入数据，满了以后才开始读入缓冲区内容到计算机内部，所有缓冲区都有这种特点



双缓冲：也就是有两个缓冲区



循环缓冲

缓冲池