操作系统复习

绪论

操作系统与计算机体系结构的关系

地位

- 不解决具体问题,不负责生成解决应用问题的程序
- 所有硬件之上,所有软件之下.是各种软件的基础运行平台

体系结构

- 应用软件
- 编译器,连接器,数据库,网络
- 操作系统
- 硬件

操作系统与各层之间的关系

- OS对各层的管理和控制
    - 管理控制硬件
        - 控制CPU的工作
        - 访问存储器
        - 设备驱动、中断处理
    - 管理控制应用软件
        - 控制管理
        - 提供用户界面和服务
- 各层对OS的制约和影响
    - 下层硬件环境的制约
        - 提供OS运行环境
        - 限制了OS的功能实现
    - 用户和上层软件的要求
        - 需求
        - 服务和洁面

存储程序式计算机特点

- 基本部件
    - CPU
    - 存储器
    - I/O

- 集中顺序过程控制
    - 过程:按照顺序来处理程序,模拟人们手工操作
    - 集中控制:由CPU集中管理
    - 顺序:有程序计数器

计算机系统结构与操作系统关系

- 多道程序设计技术
- 分时技术
- 资源分配与调度
- 计算机的顺序计算模型和操作系统的并行计算模型

操作系统的形成与发展

手工操作阶段

单道批处理

- 用户把作业交给操作员,操作员送入到输入设备上
- 常驻监督程序自动地装入程序,把控制权交给作业
- 作业执行完成后,控制权交给监督程序,监督程序继续调入后续作业.
- 作业只能按照顺序一个一个地被执行,期间不可以被打断.A执行完毕了,B才可以投入执行.

多道批处理

- 多道
    - 思路
        - 内存中同时存放几道相互独立的程序,这些程序在计算机系统中同处于开始到结束之间的状态
        - 在管理程序控制之下，相互穿插地运行，处理机和外设尽量处于忙碌状态
        - 当某道程序因某种原因不能继续运行下去时(如等待外部设备传输数据)，管理程序便将另一道程序投入运行。在多道系统里面,只有I/O程序才会处理机
    - 特点
        - 宏观上并行：多个程序同时占用多个资源同时执行
        - 微观上串行：最多一个程序占用CPU，多个程序交替占用CPU
    - 硬件支持
        - 通道技术
        - 中断技术

分时

- 把处理机时间划分成很短的时间片(如几百毫秒)，轮流分配给各个应用程序使用，如果某个程序在分配的时间片用完之前计算还未完成，该程序就暂时中断，等待下一轮继续计算。

实时

- 计算机对于外来信息能够在被控对象允许的截止期限(deadline)内作出反应。

操作系统的定义

定义

- 大型软件系统
- 计算机系统软、硬件资源的分配；
  - 操作系统会对处理机,内存,设备和文件进行管理
- 控制和协调并发活动；
- 提供用户接口，使用户获得良好的工作环境。

特征

- 并发
- 共享
- 不确定性

操作系统的基本类型

批处理系统

- 该系统把用户提交的程序组织成作业形式。作业成批送入计算机，然后由作业调度程序自动选择作业，在系统内多道运行。
- 与分时的区别:作业只有在遇到I/O请求的时候才放弃CPU控制权,而分时系统可以在时间片到来的情况下让进程放弃CPU控制权
- 多道和单道的区别
  - 多道允许多个程序在内存,单道不行
  - 单道不允许被打断,多道可以
  - 多道运行多个程序交替执行A程序执行一会儿换B程序,B程序执行一会换A程序,单道只能让一个程序顺序执行下去.

分时操作系统

- 分时操作系统是操作系统的另一种类型。它一般采用时间片轮转的办法，使一台计算机同时为多个终端用户服务。该系统对每个用户都能保证足够快的响应时间，并提供交互会话功能。
- 进程会被两个打断,一个是I/O操作,一个是时间片到了,这两个都会被打断,然后交给下一个进程处理.

实时操作系统

- 实时操作系统对外部输入的信息，能够在规定的时间内处理完毕并作出反应。
    - 类型
        - i 实时控制　生产过程控制、作战指挥
        - ii 实时信息处理 订购机票、情报检索
- 硬实时系统
    - 系统必须满足应用程序对截止期限(deadline)的要求，若错过了截止期限，将导致灾难性后果。
- 软实时系统
    - 系统中截止期限被错过的情况下，只造成系统性能下降而不会带来严重后果。

多道:用户看上去多个程序在同时运行,有多个程序同时处于开始到结束之间的状态,若干个程序存储在in诶存中,在管理程序的控制下,穿插,依次,交错着运行,这些程序在计算机系统中同时处于开始~结束的状态.
就是程序A可以先执行一会儿,然后交给程序B接着执行,接着程序A再回来接着运行

结构和硬件支持

操作系统虚拟机

在裸机上配置了操作系统程序就构成了操作系统虚拟机,操作系统在裸机上运行,用户在扩充后的操作系统虚拟机上运行

裸机的指令系统

- 机器指令

操作系统虚拟机的指令系统

- 操作命令
- 系统功能调用

操作系统的组织结构

简单结构

- 没有拆分成模块

层次结构

- 将操作系统分为多层（level）

微内核结构

- 所有的用户程序和操作系统管理模块都和一个内核相互联系

外核结构

- 让内核分配机器的物理资源给多个应用程序，并让每个程序决定如何处理这些资源

VMM虚拟机

- 虚拟机管理器将单独的机器接口转换成很多的虚拟机，每个虚拟机都是一个原始计算机系统的有效副本, 并能完成所有的处理器指令

处理机的特权级

处理机的态

- 代表中央处理机的工作状态,代表当前处理机正在执行哪些程序,决定处理机的态.
- 分类
  - 管态
    - 操作系统管理程序执行的时候
    - 可以使用全部指令、使用全部系统资源
  - 用户态
    - 用户程序执行的时候
    - 禁止使用特权指令

特权指令集

- 外部设备输入输出
- 修改特殊寄存器
- 改变机器状态

如何区分处理机的态呢?

- 利用处理器状态标志将处理器设置成不同状态
- 引入程序状态字PSW

中断技术

定义

- 某个事件发生的时候,系统中止现有程序进行、引出事件处理程序对事件进行处理,处理完毕后返回断点继续执行.

分类(了解即可)

- 功能
  - 输入输出
    - I/O
  - 外中断
    - 时钟,控制台等来自处理机外部的
  - 机器故障中断
    - 电源故障等
  - 程序性中断
    - 非法操作等
  - 访管中断
- 中断方式
  - 不是正在运行的程序期待的中断(被动打断)
  - 是运行的程序期待的事件(主动打断)
- 相关概念
  - 中断源：引起中断的事件称中断源，如打印完成中断，其中断源是打印机。
  - 断点：发生中断时正在运行的程序被暂时停止，程序的暂停点称为断点。
  - 中断响应：是处理机发现有中断请求时，中止现运行程序的执行并自动引出中断处理程序的过程。
  - 中断是硬件和软件协同的处理的,由硬件来发现中断进入中断,进入中断后,然后让软件来执行对中断事件的处理.

中断响应

- 保护现场
  - 中断发生的时候,把现场信息保存在主存中.
- 恢复现场
  - 把现场信息从主存送入到相应的寄存器
- 所以说,什么是中断响应呢?
  - 中断响应是CPU发现有中断请求,中止现行程序进行,引出中断处理程序的过程.(进入中断程序执行)
- 过程
  - 保留断点信息和现场
  - 写入PC和PS,自动进入相应的中断处理程序

软件中断处理

- 硬件完成了中断进入过程后,由相应的中断处理程序获得了控制权,进入了软件中断处理的过程.

中断返回

- 执行恢复现场,把现场信息从主存送入到相应的寄存器

中断处理的顺序

- 中断响应,保留断点,进入中断处理程序
- 中断处理程序处理中断
- 中断返回,把主存中的现场信息恢复到各个寄存器里面

多重中断

- 中断优先级
  - 给中断设置一个优先级
- 中断屏蔽
  - 可以屏蔽中断,就是屏蔽掉某些中断请求达到不进入中断的目的.
- 多个中断的处理
  - 处理中断的时候禁止被其他中断打断
  - 允许高优先级的中断打断低优先级的中断

必要的硬件支持

存储器

时钟

用户接口

用户工作环境

用户工作环境的形成

- (1) 系统提供各种硬件、软件资源
- (2) 系统设计并提供使用方便的命令集合
- (3) 系统将OS装入计算机并初始化，形成可供使用的工作环境(OS使用硬件工作环境,用户使用OS工作环境)
- 提供资源,设计命令集合,接着装载OS,用户工作环境就形成了
- 将操作系统的必要部分装入主存并对OS进行初始化,最后使操作系统处于命令接受状态

系统引导的方式

- 现场独立引导方式(滚雪球方式)
  - OS核心文件在系统本身的存储设备中
- 辅助下装模式
  - OS的主要文件在系统启动后从另外的计算机系统中获得操作系统的常驻部分

独立引导方式

- 系统加电,CPU初始化
- BIOS启动固件(恒定在某个内存区域)将磁盘重的操作系统引导程序加载到空闲区间里面,转操作系统引导程序执行(主引导记录)
- 加载程序会将操作系统的代码和数据加载到内存中,接着转操作系统运行(跳转到操作系统的起始地址)
- 接着操作系统核心初始化各种系统结构和参数
- 最后系统初始化,装在命令处理程序或者图形用户界面,并初始化.

LInux引导方式(略)

系统生成

- 就是编译并且链接出来一个操作系统
    - 所谓系统生成，就是指为了满足物理设备的约束和需要的系统功能，通过组装一批模块来产生一个清晰的、使用方便的操作系统的过程。

运行一个应用程序的过程

步骤

- 首先获得一个源程序
- 将源程序编译或者汇编成一个目标模块
- 将这个目标模块与其他需要的子程序和例行程序的目标模块连接装配在一起获得可运行文件
- 将可运行文件放入主存,启动运行

连接类型

- 静态连接
    - 一个源程序经编译后，生成一个可重定位的目标模块，并产生内部符号表和外部符号表，供连接程序 (Link)使用。
    - 内部符号表:本模块可以被其他程序调用的入口地址
    - 外部符号表:本模块要调用的外部程序模块名
    - 连接
        - 将各模块连接成为一个整体；
        - 构造全程符号表，在其中填写模块的逻辑地址；全程符号表记录了完整的用户程序里面的所有符号的逻辑地址,是每个模块的内部符号表的集合.
        - 查找各程序段的外部调用表，填入对应调用函数的地址。
- 动态连接
    - 动态连接不需要将外部函数链接到目标文件中。而是在应用程序中需要调用外部函数的地方作记录，并说明要使用的外部函数名和引用入口号。
    - DLL,动态连接库

操作系统用户界面

定义

- 操作系统的用户界面 (或称接口) 是操作系统提供给用户与计算机打交道的外部机制。用户能够借助这种机制和系统提供的手段来控制用户所在的系统。

用户界面

- 操作界面(命令界面)
    - 命令行
        - 作业控制语言(批处理系统)
            - (1) 一种命令语言，包括作业处理命令和资源请求命令
            - (2) 脱机方式下系统提供作业控制语言
        - 键盘命令(分时操作系统)
            - 为联机用户提供的操作命令,用户通过这一组命令直接控制和干预程序执行,现代操作系统的命令行命令.
    - 可视化图形系统
- 系统功能服务界面(程序接口)

系统功能调用

操作系统提供实现了各种功能的例行子程序

调用操作系统服务功能的方法

- 统一进管的方式(使用访管指令调用系统调用)[svc n],程序调用svc指令即可调用操作系统的系统调用,n代表调用什么样的系统调用.
- 然后当处理机执行到访管指令时发生中断，该中断称为访管中断，它表示正在运行的程序对操作系统的某种需求。

系统调用的顺序

- 首先用户程序调用svc命令
- 启动访管中断
- 进入访管中断的处理程序
- 接着在中断处理程序里面根据n的值转移到不同的例行子程序.执行对应的例行子程序.例行子程序会执行用户程序请求的系统调用
- 例行子程序执行完毕回到访管中断,再回到用户程序
- 用户程序->访管中断处理程序->例行子程序.(函数调用顺序)

系统调用的特点

- 每个系统调用对应一个系统调用号
- 每个系统调用有一个对应的执行程序段
- 每个系统调用要求一定数量的输入参数和返回值
- 整个系统有一个系统调用执行程序入口地址表

进程管理

进程的引入

顺序程序

- 程序的一次执行过程称为一次计算,它由许多简单的操作组成
- 一个计算中的若干操作必须按照严格的先后顺序执行,在一次计算的执行中,计算中的各个操作要一个操作一个操作地依次执行.同样,计算也是一次执行的,只有前面的计算做完了,下一个计算才能接着做
- 顺序性- 处理机的操作严格按照程序所规定的顺序依次执行
- 封闭性- 程序一旦开始执行,就不会收到外界因素的影响
- 可再现性- 程序执行的结构和它执行的速度无关

并发程序

- 若干个程序段同时在系统中运行，这些程序段的执行在时间上是重叠的，一个程序段的执行尚未结束，另一个程序段的执行已经开始，即使这种重叠是很小的一部分，也称这几个程序段是并发执行的。
- 并发语句记号:
  cobegin
  S1,S2,S3...Sn
  coend
- 特点
  - 失去封闭性和可再现性-两个并发程序若干操作的先后顺序
  - 一个程序可以对应多个计算
  - 多个计算之间会有并发执行的相互制约

进程概念

定义

- 所谓进程，就是一个程序在给定活动空间和初始环境下,在一个处理机上的一次执行过程。
  就是程序的一次执行就叫做进程.

特点

- 动态性
  - 动态性:进程可以动态创建,也可以动态结束
  - 并发性:可以被调度并占处理机运行
  - 独立性:不同进程互不影响
  - 制约性:不用的进程因访问共享资源进程间同步而制约

进程与程序的区别

- ① 程序是静态的概念(存储在内存/外存的代码)，进程是动态的概念(程序在处理机运行的过程,是运行中的程序)；
- ② 进程是一个独立运行的活动单位；
- ③ 进程是竞争系统资源的基本单位；
- ④ 一个程序可以对应多个进程，一个进程至少包含一个程序。

进程的基本状态

- 状态转换图(要记得)
- 三种基本状态
  - 运行:该进程已获得运行所必需的资源，它的程序正在处理机上执行。
  - 等待:进程正等待着某一事件的发生而暂时停止执行。这时，即使给它CPU控制权，它也无法执行。
  - 就绪:进程已获得除CPU之外的运行所必需的资源，一旦得到CPU控制权，立即可以运行。
- 转移
  - 就绪->执行:进程调度即可就绪状态转为运行状态
  - 等待->就绪:处于等待状态的进程中相关服务完成或者相关资源获得完成
  - 运行->等待:进程提出某种服务请求,比如说I/O
  - 运行->就绪:只有分时有,时间片到
  - 等待状态由于需要等待某种服务完成,所以说不可以被进程调度调用.
- 进程基本状态的拓展
  - 拓展1:程序执行完了,进程还可以被回收.
  - 拓展2:添加挂起操作,添加静止状态,静止表示当前进程不在主存里面,在虚存里面.
  - Unix进程,运行态分成用户态运行和核心态运行.

进程描述

- 进程的组成
  - 进程控制块:描述进程与其他进程、系统资源的关系,以及进程所处的状态
    - 进程标志性
    - 进程的状态
    - 进程队列的指针next
    - 进程优先级,家族联系,现场保护区
  - 进程内部的程序和数据
- 进程的组织
  - 就绪队列,所有处于就绪状态的队列
  - 等待队列,有多个,每个队列表示所有因为同个某种原因而等待的进程
  - 运行指针,当前是什么进程正在运行

进程控制

进程创建

- create (name，priority)
  - name为标识符
  - priority是优先级
- 实现方法
  - 查PCB总链,是否出现同名现象.
  - 向系统申请一个空PCB,没有空PCB就退出
  - 将入口信息填入PCB
  - PCB入就绪队列

进程撤销

- kill(),exit()
- 实现方法
  - 由运行指针获得当前进程pid
  - 释放本进程资源给父进程
  - 从总链摘下
  - 释放PCB结构

进程等待

- susp(chan)
  - chan:进程等待的原因
- 实现方法
  - 保护CPU现场
  - 置该进程为等待状态
  - 将该进程的PCB结构插入到对应的等待队列中

进程唤醒

- wakeup(chan)
  - chan:进程等待的原因
- 实现方法
  - 找到chan对应的等待队列
  - 将首进程移除等待队列,将该进程插入到就绪队列,改变状态

进程的相互制约关系

临界资源

- 一次只允许一个进程使用的资源
  - 硬件:输入机,打印机,磁带机
  - 软件:公共变量

临界区

- 对于公共变量(存储器)修改的程序段

进程互斥

- 在操作系统中，当某一进程正在访问某一存储区域时，就不允许其他进程来读出或者修改存储区的内容

进程同步

- 并发进程在一些关键点上可能需要互相等待与互通消息， 这种相互制约的等待与互通消息称为进程同步。

进程同步机构

锁

- 用一个变量w代表某种资源的状态
  - w=1,资源被占用
  - w=0,资源没有被占用
- 上锁原语lock(w)
  - 执行到lock的时候判断w是多少
  - 如果w=1,就被阻塞,进程无法往下继续运行,直到什么时候w=0.
  - 如果w=0,不会被阻塞,进程可以继续执行,并且会把w赋值为1
- 开锁原语unlock(w)
  - 执行到unlock的时候,把w赋值为0即可.

信号灯

- 信号灯是一个确定的二元组 (s，q)，s是一个具有非负初值的整型变量，q是一个初始状态为空的队列。操作系统利用信号灯的状态对并发进程和共享资源进行控制和管理。在处理信号灯操作的时候,就是对那个整型变量进行处理.
- 一般来说就是声明一个变量,在进行P-V操作的时候就把信号灯变量当作参数传进P-V操作里面.P-V操作就会更改变量的值
- P操作
  - 先把信号灯变量的值-1
  - 如果相减的结果为负,进程阻塞,插入到信号灯的对应的等待队列里面
  - 相减的结果非负,继续运行
- V操作
  - 信号灯变量值+1
  - 信号灯的结果不为正,则从信号灯等待队列中取出一个元素放入就绪队列
- 信号灯数值的意义
  - s>0,表示还可以允许多少个进程访问资源
  - s<0,|s|表示有多少个进程正在等待

进程互斥与同步的实现

进程互斥的做法

- 对于每一个锁,进入临界区之前上锁,离开临界区的时候解锁即可(回忆,临界区就是对公共变量进行处理的代码段)
- 对于一个临界区有一个临界信号灯管理,进入临界区就P(s),离开临界区就V(s)
- 语法检查
    - 信号灯的初值为非负整数
    - 除初始化外，仅能由P、V原语对信号灯进行操作
    - P、V操作一定成对出现
    - P操作在前，V操作在后

两类同步问题的解法

- 进程流图
    - 表示进程之间执行的先后次序,某些进程的完成代表某些进程可以开始执行的顺序
    - 可以用多个信号灯表示,在每个进程前面加上若干个P操作,在每个进程后面加上若干个V操作
    - 其中V操作用于通知其他进程本进程已经执行完毕(相当于消息发送者)
    - P操作用于接受上一层V操作发送来的消息
    - 假如说有一个关系p1->p2,p1执行完了才能执行p2,那么就有一个信号灯,初值为0,在p2的开始加上P操作,p1的末尾加上V操作,有多少对这样的关系就有多少个信号灯
- 共享缓冲区
    - 问题概述:有两个进程,一个负责读,一个负责写
    - 可以转化成进程流图,只有读完了,才能写,只能写完了,才能读.则规定两个信号灯,一个在读进程头P操作,在写进程尾V操作,一个在写进程头P操作,在读进程尾V操作.
      当然还可以用两个信号灯表示空闲数和占用数,占用数大于0,可以读取,空闲数大于0,可以写
    - 写进程
        - p(sb)
            - sb代表缓冲区的空位置数,初值为1
        - 数据放入buf
        - v(sa)
            - sa代表缓冲区的数据数,初值为0
    - 读进程
        - p(sa)
        - 取数据
        - v(sb)
- 生产者消费者
    - 若干个进程往一定量的容器里面放东西,若干个进程往一定量容器里面写东西
    - 信号灯设置
        - 两个同步信号灯——

        - sb： 表示空缓冲区的数目，初值 = n
        - sa： 表示满缓冲区 (即信息)的数目，初值 = 0
    - 一个互斥信号灯
        - mutex：表示有界缓冲区是否被占用，初值 = 1
- 生产者:
    - p(sb)表示空缓冲区少1,如果没有的话就阻塞
    - p(mutex)进入临界区
    - v(mutex)退出临界区
    - v(sa)写完,信息数+1
- 消费者:
    - p(sa)表示信息数-1
    - v(sb)读完,空闲区+1
- 玩意生产者或者消费者不愿意等待了怎么办?
    - 可以用一个共享变量p来表示剩余资源或者是占用的资源数量.对这个p用P-V操作保护,用if-else语句表达判断,如果不可以就直接释放进程.而不是P操作.

Unix/Linux关于进程的系统调用

fork()

- 创建一个新的进程,父子两个进程共享fork()后面的代码,父子进程同时从fork()函数返回返回值开始一块往下运行.
- 对于父进程,函数返回值是子进程的pid,对于子进程,函数返回值是0,然后父子进程并行投入运行,都从fork()函数返回返回值开始继续往下运行

exit()

- 该进程结束执行

wait(&pid)

- 该进程需要等待号码为pid的进程结束才继续执行

P操作

本质上会让某种元素-1

并且某种元素数量不为正数的话就不能运行

V操作

本质上会让某种元素+1

还有一点,关于变量的处理不一定需要PV操作,可以加临界区和if语句进行控制

mutex

只要用临界区就在临界区的首尾加上P和V

资源管理概述

资源管理概述

资源管理功能

- 数据结构描述
- 调度原则
- 资源分配的实现
- 安全保护

分配类型

- 静态分配
  - 在进程运行前,操作系统一开始就把他所要的全部资源都分配
- 动态分配
  - 进程在运行的过程中,边运行边向操作系统提出申请,操作系统根据申请分配资源
- 宏观上
  - 作业调度
- 微观上
  - 进程调度

虚拟资源

- 物理资源 (实资源)
- 虚拟资源 (逻辑资源)

资源分配的机构和策略

资源分配的机构

- 资源描述器
  - 描述各类资源的最小分配单位的数据结构
  - 资源名,类型,大小,地址,分配标志,描述器连接信息,存取权限等
- 资源信息块
  - 等待队列头指针>请求者队列
  - 可利用资源队列头指针
  - 资源分配程序入口地址

资源分配策略

- 先请求先服务
  - 队列结构:每一次新产生的请求在队尾,
  - 当资源可用的时候取队首元素,并满足其需要.
  - 先来的请在队列首部,后来的在队列末尾

- 优先调度
    - 对每一个进程指定一个优先级；
    - 每一个新产生的请求，按其优先级的高低插到相应的位置,优先级越高,月靠近队首,优先级越低,月靠近队尾.
    - 当资源可用时，取队首元素，并满足其需要。
- 针对设备特性的调度策略
    - 磁盘
        - 磁盘的工作原理
            - 先移动磁盘的移动臂找到合适的柱面
            - 接着旋转这个柱面,找到要找的磁道和扇区
        - 调度策略
            - 移臂调度
                - 有多个I/O请求,怎样移动磁盘的移动臂来满足?
                - SCAN:总是选取与当前移动臂前进方向上最近的那个I/O请求
                - SSTF:总是选取与当前移动臂位置距离最短的那个I/O请求
            - 旋转调度
                - 总是选取与当前读写头最近的那个I/O请求
            - 先执行移臂调度在执行旋转调度

死锁

定义

- 在两个或多个并发进程中，如果每个进程持有某种资源而又都等待着别的进程释放它或它们现在保持着的资源，否则就不能向前推进。此时，称这一组进程产生了死锁。
- 进程在占有某个资源而请求某种资源,当该进程占有的资源是别人请求的资源时,就可能产生死锁

资源分配图

- 两类顶点
    - 所有的进程P
    - 所有的资源R
- 两类有向边
    - 资源请求边
        - 进程P请求资源R,P->R
    - 资源分配边
        - 资源R分配给进程P,R->P

产生死锁的必要条件

- 互斥条件
  - 涉及的资源是非共享的,为临界资源
  - 一个资源不能被两个进程同时获得
- 不剥夺条件
  - 进程所获得的资源在未使用完毕之前,不能被其他进程强行夺走.
- 部分分配
  - 进程每次只申请他所需要的资源的一部分,在等待新一批资源的时候,进程继续占用分配到的资源
  - 进程每次只申请他需要资源的一部分,资源一旦分配给它,进程就会持续占有这一部分资源
- 环路条件
  - 存在一种进程的循环链，链中的每一个进程已获得的资源同时被链中下一个进程所请求。

解决死锁问题的策略

- 把四个必要条件的其中一个弄成不满足
- 死锁预防
  - 静态预防死锁
    - 在作业调度的时候就给选中的作业分配它所需要的全部资源,(一开始就分配所有资源给作业)
  - 有序资源分配法
    - 系统中所有资源都给定一个唯一的编号，所有分配请求必须以上升的次序进行。当遵守上升次序的规则时，若资源可用，则予以分配；
- 死锁避免
  - 要求进程声明需要资源的最大数目,再分配资源的时候判断是否会出现死锁,只有在不会死锁的时候才分配资源
- 死锁恢复
  - 破坏循环等待
    - 杀死有关进程和删除文件
- 定理
  - P个进程共享m个同类资源,如果所有进程对资源的最大需要数目之和小于p+m,这个就不会发生死锁.

系统状态分析

- 初始状态
  - 一组确定的进程集合
  - 一组不同类型的资源,以及各类可利用的资源数目
  - 资源分配矩阵:a_{ij}代表进程i占有j类资源数目
  - 资源请求矩阵:d_{ij}代表进程i还需要j类资源数目
- 安全状态

- 当进程请求某类资源时，进程对该类资源的需求量小于当前时刻系统所拥有的该类资源的数目，那么满足进程的这次请求，系统是安全的。
- 要求进程声明需要资源的最大数目(进程最多最多就需要那么多资源),如果当前计算机剩余的资源大于进程声明所需要的总的资源.就称为安全的..

银行家算法

- 申请者事先说明对各类资源的最大需求量。在进程活动期间动态申请某类资源时，由系统审查现有该类资源的数目是否能满足当前进程的最大需求量，如能满足就予以分配，否则拒绝。
- 当前剩余资源>该进程声明的最大需求量,就可以满足该进程的局部资源分配需求.

处理机调度

处理机的多级调度

确定数据结构

制定调度策略（调度原则）

给出调度算法

具体的实施处理机分派

作业调度

作业调度的内容

- 对存放在辅存设备上的大量作业，以一定的策略进行挑选，分配主存等必要的资源，建立作业对应的进程，使其投入运行。
    - 作业存储在辅存设备上
    - 在辅存中挑选一个作业,将其载入到主存.
    - 作业被载入到主存后,建立其对应的进程结构,使之投入运行.
    - 运行完毕,回到第二步

作业控制块JCB

- 存放作业控制和管理信息

作业调度算法的性能衡量

- 周转时间
    - 各作业提交给计算机系统到该作业的结果返回给用户所需要的时间。(从作业提交给计算机(存放在辅存)到结果返回给用户的时间)
    - 简单来说,可以理解为作业在磁盘中放置的时间+作业在主存中被运行的时间
- 带权周转时间
    - 作业周转时间/作业实际运行时间

- (作业运行时间+作业在磁盘中等待被执行的时间)/作业运行时间

作业调度算法

- 先来先服务
- 选择一个运行时间最短的作业调入内存
- 响应比高者优先:
  - 响应时间(执行时间+作业等待了时间)/执行时间
- 优先调度算法
- 均衡调度算法

进程调度

对进入主存的所有进程，确定哪个进程在什么时候获得处理机，使用多长时间。

分成两种功能

- 调度
  - 组织和维护就绪进程队列
- 分派
  - 处理机空闲的时候,从就绪队列首部选取一个PCB投入运行

分成两种调度方式

- 非剥夺方式
- 剥夺方式

调度算法

- 进程优先数调度算法
  - 优先级最高的先被调度
  - 优先数可以是静态确定,还可以是动态确定,可以在系统运行的途中边运行边动态地调整进程的优先数.
- 循环轮转调度算法
  - 每个进程被调度到后，占用一个时间片，时间片用完后，该进程让出CPU，排在就绪队列的队尾。多个进程循环轮转。
  - 队列结构
    - I/O等待队列
    - 低优先队列:时间片到,进入低优先队列
    - 高优先队列:等待转就绪:高优先
    - 下一个进入的,只有高优先队列什么都没有,

Unix进程调度

Unix系统采用优先数调度算法

优先数计算公式

- p_pri = min{127, (p_cpu/16-p_nice+PUSER)}
- p_cpu 进程占用CPU的程度R1:
  - R1=Tu/(Tu+Tnu)
    - Tu:进程创建后占用CPU的累计值
    - Tnu：进程生成后没有占用CPU的累计值
- p_nice 用户通过系统调用nice()设置的进程优先数
- PUSER 常数，其值为100

优先数低的,优先级越高,就优先考虑该进程执行

内存管理

主存管理概述

存储器：能接收和保存数据、并根据命令提供这些数据的装置，分为主存和辅存。

- 主存
  - 内存
- 辅村
  - 磁盘

主存共享方式

- 大小不同
  - ① 分区存储管理
  - ② 段式存储管理
- 大小相等的区域
  - 页式存储管理
- 段页式存储管理

逻辑组织

- 一维地址结构
  - 一个程序是一个连续、线性的地址结构；
  - 确定线性地址空间中的指令地址或操作数地址只需要一个信息。
  - 可以把程序的地址结构近似成一维数组,是一段连续的线形分布的空间,只需要一个信息就可以确定地址.
- 二维地址结构

    - 一个程序由若干个分段组成，每个分段是一个连续的地址区
    - 确定线性地址空间中的指令地址或操作数地址需要两个信息，一是该信息所在的分段，另一个是该信息在段内的偏移量。

主存管理的功能

几个概念

- 物理地址是计算机主存单元的真实地址，又称为绝对地址和实地址
- 物理地址的集合所对应的空间组成了主存空间。
- 用户的程序地址 (指令地址或操作数地址)均为逻辑地址。
- 程序地址空间:用户程序所有的逻辑地址集合对应的空间。

主存管理功能

- 实现逻辑地址到物理主存地址的映射
    - 逻辑地址变换成主存中的物理地址的过程,称为地址映射.
    - 地址映射的时机
        - 编译的时候确定地址映射的关系
        - 在程序装入的时候确定地址映射关系,静态地址映射:程序代码的逻辑地址和存储空间的映射在程序装入内存的时候就已经确定了,由重定位装入程序确定
        - 在程序运行时确定地址映射关系:在程序执行期间，随着每条指令和数据的访问自动地连续地进行地址映射，这种地址变换方式称为动态地址映射。(重定位寄存器)
- 主存分配
    - 主存资源信息块：等待队列；空闲区队列；主存分配程序
- 存储保护
    - 主存按照区的模式分配给各用户程序使用,每个用户程序必须在给定的存储区域内活动即可.
    - 上下界保护
        - 有上界寄存器和下界寄存器,程序访问内存只能使用在上界寄存器和下界寄存器之间的区域(物理地址)
    - 基址、限长寄存器保护
        - 有限长寄存器,和基址寄存器,基址寄存器表示程序在内存中存储空间从何开始,限长寄存器限制程序访问的逻辑地址,逻辑地址<限长寄存器的值
- 主存扩充
    - 可行性
        - 局部性特征
            - 时间局部性: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个较短时期内
            - 空间局部性：当前指令和邻近的几条指令，当前访问的数据和邻近的几个数据都集中在一个较小区域内
            - 分支局部性：一条跳转指令的两次执行，很可能跳到相同的内存位置
    - 实现方法
        - 程序的全部代码和数据存放在辅存中；

        - 将程序当前执行所涉及的那部分程序代码放入主存中；
        - 程序执行时，当所需信息不在主存，由操作系统和硬件相配合来完成主存从辅存中调入信息，程序继续执行。
    - 虚拟存储器
        - 由OS和硬件相配合完成主存和辅存之间信息的动态调度,这样子好像OS提供了一个存储容量比实际主存大得多的存储器,这个存储器称之为虚拟存储器
            - 逻辑地址与物理地址分开
            - 存储空间与虚地址空间分开
            - 提供地址变换机构
        - 用户通过逻辑地址访问虚拟存储器,接着OS和硬件动态调度,将虚拟存储器的逻辑地址转化为物理地址来访问实际的主存

分区存储管理

静态分区

- 把内存预先划分成多个分区，分区大小可以相同或不同
- 分区个数固定，分区大小固定
- 一个分区装入一个作业
- 主存分区是如何的已经在运行前就已经确定了

动态分区

- 在运行程序的过程中:建立分区,依照用户请求的大小分配分区

分区分配所需要的数据结构

- M_RIB
    - 主存资源信息块
        - 等待队列头指针
        - 空闲区队列头指针
        - 主存分批程序入口地址
- 分区描述器(PD)
    - flag为0为空闲区
    - flag为1是已占用区
    - size是分区大小
    - next:如果是空闲区,那就是下一个空闲区的首地址,如果是已分配区,这一项为0

分区回收的思路

- 检查释放分区的在主存中的连接情况
- 如果上下邻接空闲区,则合并,成为一个新的空闲区
- 若回收分区不与任何空闲去相邻接,建立一个新的空闲区,加入到空闲队列.

选择空闲区的放置策略

- 首次适应算法:是将输入的程序放置到主存里足够装入它的 地址最低的 空闲区中。(地址最低最好)
  - 放入一个能放入这个程序的空闲块,空闲块地址越低越好.
  - 空闲区队列:空闲区地址由低到高排序
- 最佳适应算法:最佳适应算法是将输入的程序放置到主存中与它所需大小最接近的空闲区中.
  - 放入一个能放入这个程序的空闲块,空闲块大小越小越好.
  - 空闲区队列:由小到大排序
- 最坏适应算法是将输入的程序放置到主存中与它所需大小差距最大的空闲区中,
  - 放入一个能放入这个程序的空闲块,空闲块大小越低越好.
  - 空闲区队列:由大到小排序

碎片问题

- 在已分配的区域里面存在着一些没有被充分利用的空闲区.
- 解决技术
  - 拼接技术
    - 移动存储器中某些已分配区中的信息,使得本来分散的空闲区连接成比较大的空闲区
  - 对换技术
    - 选择内存中的某个进程暂时移出到磁盘，腾出空间给其他进程，同时把磁盘中的某个进程换进主存使其投入运行
  - 伙伴系统
    - 把一个大的存储块分为大小相等的两个小存储区（互为伙伴，大小均为2的K次幂）
    - 分配过程
      - 由小到大在空闲块数组中寻找到最小的可用空闲块.
      - 如果空闲块过大,就用可用空闲块二等分,直到得到合适的可用空闲块
      - 空闲块的大小只可能是2的n次方.若程序大小是s,且满足$2^{k-1}<s\leq 2$k,那么就把大小为2^k次的分区分配
    - 释放过程
      - 把释放的块加入空闲块数组
      - 合并满足合并条件的空闲块
        - 大小相同
        - 地址相邻
        - 且低地址空闲块是2^{i+1}的位数

页式存储管理

基本概念

- 程序的地址空间(虚拟地址空间)被等分成大小相等的片,称为页面,又称为虚页
- 主存(物理地址空间,实地址空间)又被分成大小相等的片,称为主存块,又称为实页
- 为了实现从地址空间到物理主存的映象，系统建立的记录页与内存块之间对应关系的地址变换的机构称为页面映像表，简称页表。(程序逻辑地址中虚页号与物理主存中实页号之间的变换映射)
    - 程序逻辑地址有虚页号,内存的物理地址里面为实页号,程序就通过虚页号替换成实页号来找到程序或数据实际的位置.
    - 总的来说,是记录页和块之间对应关系的地址变换的结构

页式地址变换

- 虚地址(程序逻辑地址)结构
    - 其中高位为页号P
    - 低位为页内偏移
- 地址变换过程
    - CPU给出操作数
    - 分页机构自动把逻辑地址分成两部分高地址页号为P,低地址页内偏移为W
    - 根据页表始址寄存器指示的首地址PTBR,加上页号P,就是该页号对应的页表项的地址(PTBR+P就是这个页对应的页表项地址),获得物理块块号
    - 将块号B和页内偏移量并在一起,就形成了主存地址(物理地址)
- 快表TLB
    - 先在快表中查找有没有相关页表项记录,快表是一个独立的硬件,独立于内存之外
    - 如果快表中没有,只能查找存储在内存中的页表,然后把查出来的页表项记录在快表里面
- 多级页表
    - 间接引用
    - 页表项中可能存储的不是物理块号,而是下一级页表的首地址

请调页面的机制

- 简单页式系统
    - 装入一个程序的全部页面才能投入运行。
- 请求页式系统
    - 装入一个程序的部分页面即可投入运行。
- 扩充页表
    - 加一个中断位,表示此页是不是在主存里面,如果是0表示在主存,如果为1表示不在主存
- 缺页中断
    - 缺页中断就是要访问的虚页(逻辑地址对应的虚页)不在主存，需要操作系统将其调入主存后再进行访问。
    - 缺页中断的处理
        - 如果没有空闲块(物理内存对应的实页),则需要选一个虚页淘汰,这个虚页对应的主存块重新分配.(一个程序只会分配若干个空闲块(内存空间),如果不够就要淘汰对应一页中的一块)一般来说就是选择一个页表的页表项中断位为1

对应的主存块分配给另外一页.
        - 然后从外存中调入所需的页
        - 调整页表
    - 抖动
        - 简单地说，导致系统效率急剧下降的主存和辅存之间的频繁的页面置换现象.
- 淘汰策略
    - 选择淘汰掉哪一页的规则叫做置换策略,决定淘汰哪一页
    - 一般来说淘汰出现在分配的主存块数比程序逻辑地址页数少的情况下
    - 具体的淘汰策略
        - FIFO算法
            - 总是选择在主存中居留时间最长的那一页
            - 实现的数据结构
                - 页号表
                    - 用一个变量k来表示下一次替换哪一页,用一个数组p记录页号,要替换的页=p[k],然后p[k]=新的页号,k++
                - 存储分块表
                    - 构建一个队列,队列的每个元素都是(页号,块号)
                    - 每次选一个出队,出队元素对应的页号淘汰
                    - 出队元素的页号替换成新分配页面的页号,块号不变,再进入队尾
        - LRU(最久未使用)算法
            - 软件方法：采用页号栈
                - 页号对应的是一个栈
                - 假如说访问了某个页,这个页号进入栈底
                - 要淘汰某个元素了,栈顶元素出栈(栈顶元素对应的页号被淘汰),新的页号放入栈底

段式及段页式存储管理

段式地址空间

- 由若干个逻辑分段组成，每个分段有自己的名字，对于一个分段而 言，它是一个连续的地址区。
- 段:程序中一组逻辑意义完整的信息集合.
- 段式地址变换
    - 取出程序地址(s，w)；其中s为段号,w为段内位移
    - 用s检索段表；找到该段的起始地址B
    - 如w＜0或w≥L则主存越界；
    - (B＋w)即为所需主存地址

段页式系统

- 段页式地址变换
  - 首先还是一样程序地址(s,p,w)其中s为段号,p为该段内的页号,w为页内偏移
  - 首先根据s的数码找到这个段的页表始址PTEP
  - 根据物理地址和页内偏移找到最后的主存地址

设备管理

设备管理概述

分类

- 存储设备
- 输入输出设备
- 通信设备

特征

- 速度差异大
- 传输单位不同
- 容许的操作种类不同
- 出错条件不同

目标

- 提高设备利用率
- 方便用户使用

设备管理功能

- 状态追踪
- 设备分配和回收
  - 静态分配
    - 程序进入系统的时候就进行分配,退出系统的时候回收全部资源
  - 动态分配
    - 进程提出设备申请的时候仅从分配,使用完毕立即收回
- 设备控制

设备独立性

- 用户在程序中使用的设备与实际使用的设备无关,也就是说在用户程序中,只使用逻辑设备名.
- 逻辑设备名
  - 用户自己指定的设备名,可以更改
- 物理设备名

    - 系统提供的设备的标准名称,是永久的,可以更改的
- 系统可以根据设备的使用情况,动态地分配给程序中某类设备中的任一一台物理设备,程序都可以正确运行.
    - 也就是说,程序使用一个逻辑设备,这个逻辑设备和不同的物理设备绑定,都可以正常工作.运行途中逻辑设备A无论和物理设备B和还是物理设备C绑定都不会影响程序运行
- 在输入/输出信息的时候,信息可以从不同类型的设备上输入输出

设备控制块

- 记录设备的硬件特性,连接和使用情况的一组数据
- 设备名
- 设备属性
- 命令转换表
    - 包括特定的I/O例程地址,能表示设备能执行何种I/O操作

缓冲技术

定义

- 两种不同速度的设备之间传输信息的平滑传输过程.

为什么要缓冲?

- 处理数据流生产者与消费者速度差异
- 协调传输数据大小不一致的设备
- 应用程序拷贝语义(什么玩意?)

单缓冲操作

- 读设备得数据
    - 首先获得一个空的缓冲区
    - 设备会把物理记录(数据[从设备来的])送到缓冲区中
    - 用户请求数据时,系统将依据逻辑记录特性从缓冲区提取数据 并 发送到用户进程存储区
    - 如果用户请求数据缓冲区又没有怎么办?等待!
- 写数据到设备
    - 首先获得一个空的缓冲区
    - 将一个逻辑记录(数据[从应用来的])从进程存储区送到缓冲区中
    - 如果缓冲区写满了,系统将缓冲区内容作为物理记录写到设备上.
    - 缓冲区还是满,进程企图输出信息的时候,它需要等待

双缓冲操作

- 两个缓冲区
- 数据输入
    - 输入设备首先填满buf1
    - 进程从buf1提取数据的时候,输入设备填满buf2.当缓冲区一个空,一个满的时候就可以交换.
    - 总的来说,就是进程提取一个缓冲区,设备往另外一个缓冲区输入数据,如此循环往复.
- 数据输出
    - 首先进程填满buf1
    - 设备从buf1提取数据时候,进程就往buf2输出数据,当缓冲区一个空,一个满的时候就交换.

环形缓冲

- 缓冲区构成一个环形链表,有读指针和写指针,读写元素分别从读指针和写指针写数据

UNIX缓冲区管理

- 组成
    - 缓存数组
    - 缓存首部
        - 设备号,使用该缓冲区的设备号
        - 块号
        - 指向数据区域的指针
        - b链和av链的指针
    - 队列结构
        - 与某类设备有关的所有缓冲区称为设备缓冲区队列,简称b链.
        - 空闲缓冲区组成的队列称为空闲缓冲区队列,简称av链
- 缓冲管理算法
    - 当一块buf被分配用于读/写某设备时:从av链链首部上取下来,放入b链中.
    - 读写结束后,留在b链上,送入av链链尾
    - 进程需要的信息在buf中:在这个设备的b链上找到,然后从av链摘除,进程在读取完buf内部的信息之后,又送入av链链尾.
- 预先缓存
    - 用户进程要读数据,磁盘先把数据送入到高速缓存中,接着用户进程从高速缓存读取数据
- 延迟发送
    - 用户进程要写数据,用户进程先把数据送入到高速缓存中,磁盘从高速缓存中读取数据

设备分配

分配原则

- 静态分配:在作业载入的时候就把所有这个作业需要的资源
- 动态分配:在作业(或进程)运行的过程中,需要设备时系统根据某种分配原则分配

分配算法

- 先来先服务
- 优先级高者优先

分配策略

- 独享分配:分配独享设备:在一个作业整个运行期间占用的设备
- 共享分配:分配多个作业、进程共同使用的共享设备
- 虚拟分配
    - 所谓虚拟技术，是在一类物理设备上模拟另一类物理设备的技术，是将独占设备转化为共享设备的技术。
    - 通常把用来代替独占型设备的那部分外存空间 (包括有关的控制表格)称为虚拟设备。
    - 进程先把元素写入位于磁盘中的虚拟设备
    - 然后虚拟设备分配管理器再把磁盘中的虚拟设备数据写入物理设备
- SPOOLING(一种实例虚拟设备分配策略)
    - 预输入
        - 应用程序需要数据之前,OS已经把所需要的数据放入输入井中存放,应用程序可以直接从输入井获取数据
    - 缓输出
        - 应用程序执行的时候,将输出数据写入输出井中,当应用程序执行完毕后,OS将输出井的数据输出
    - 利用通道和中断技术，在主机控制之下，由通道完成输入输出工作。系统提供一个软件系统(包括预输入程序、缓输出程序、井管理程序、预输入表、缓输出表)。它提供输入收存和输出发送的功能，使外部设备可以并行操作。这一软件系统称为SPOOLING系统。
    - 基础
        - 辅存空间
        - 通道和中断
        - 数据结构
        - 软件
            - 预输入,缓输出,井管理程序

I/O控制

输入输出控制方式(组原提到了/不表)

I/O子系统

- 在应用程序提供I/O应用接口
- 每个通用设备类型都通过一组标准函数(以及接口)来访问.

设备处理程序

- 直接控制设备运转的程序,每一类设备都有一个设备处理程序,能同时控制同类多台设备工作
- 取一个I/O请求块
- 启动I/O操作,设备处理进程进入等待状态
- 等待I/O操作完成,设备处理进程进入就绪状态
- 设备处理进程重新运行,判断出错了吗?如果没有的话就唤醒请求此I/O操作的进程

控制I/O模块的方式

- 设备处理程序对应的设备处理进程
- 将设备与文件一样对待

I/O接口程序

- 首先把逻辑设备转化为物理设备
- 合法性检查,这个设备能否执行这个操作
- 形成I/O请求块,发送给设备处理进程

处理顺序

- 用户进程请求IO
- 首先进入I/O过程
- 由I/O过程进入I/O处理进程
- I/O处理进程启动I/O设备进行I/O操作,进入等待状态
- I/O设备执行完I/O操作后进入中断唤醒I/O处理进程
- I/O处理进程则唤醒调用该I/O的用户进程

构造

用户进程

I/O接口程序(I/O过程)

I/O处理进程

I/O设备I/O操作

文件系统

文件系统的基本概念

文件的概念

- 裸机上具有完整意义的信息集合,有一个文件名作为标识
- ① 文件是具有符号名的信息(数据)项的集合
- ② 文件是具有符号名的记录的集合

基本单位

- 信息项
- 记录

文件分类

文件的属性

- 文件名:每个文件有一个给定的名字，这个名字是由串描述且由文件内容来表示，包括文件符号名和内部标识符。(给定一个名字来唯一标记文件.)
    - 用户使用文件符号名来标记文件
    - 系统使用内部标志符来标记文件
- 文件拓展
    - 标记文件的使用特征
- 文件属性
    - 属性字:文件类别、保护级

文件系统

- 文件系统是操作系统中负责管理和存取文件信息的软件机构。
- 组成
    - 数据结构
    - 管理程序
- 功能
    - 用户视角
        - “按名存取”的功能
    - 系统视角
        - 辅存空间管理
        - 文件集合管理
        - 文件保护

文件组织两种结构

- 逻辑结构(用户角度)
- 物理结构(系统角度)
  - 在物理存储器上的表现形式
- 逻辑记录
  - 文件中按信息在逻辑上的独立含义来划分的信息单位,对文件进行存取操作的基本单位.在用户视角上面的一个文件.用户视角下的一个文件.
- 物理记录
  - 在存储介质上,由连续信息组成的一个区域称为磁盘块,也可以叫物理记录

文件的逻辑结构与存取方法

文件的逻辑结构

- 流式文件
  - 流式文件是相关的有序字符的集合，是无结构的。(仅仅是一堆字节组成的字符的集合)
  - 流式文件是按信息的个数或以特殊字符为界进行存取的。
- 记录式文件
  - 记录式文件是一种有结构的文件。这种文件在逻辑上总是被看成一组连续顺序的记录的集合。
- 变长记录
  - 每个逻辑记录大小会变化
- 定长记录
  - 每个逻辑记录大小不变

文件存取方法

- 顺序存取
  - 后一次存取总是在前一次存取的基础上进行的。
    只有取完第一个才能取第二个
- 随机存取
  - 用户以任意次序请求某个记录。
    可以随便取第n个元素

文件的物理结构

连续文件

- 一个文件分配在磁盘连续区域的物理块
- 文件在文件目录里面需要提供文件文件符号名,存放文件的第一个磁盘块块号,还有文件占据了多少个磁盘块

串联文件

- 就是文件结构是按顺序串联的块组成的,文件的信息存在若干个物理块中,每个物理块做末一个字作为链接字用来指示后续物理块的物理地址.
- 文件在文件目录里面需要提供文件文件符号名,存放文件的第一个磁盘块块号,剩下的磁盘块号可以通过每个磁盘块的链接字来指示.
- 文件分配表FAT
    - 把链接指针按顺序集中存放，构成盘文件映射表/文件分配表(FAT),串联文件的链接字不再每块里面,而是在一个统一的表里面,其中FAT的第n个元素为m代表:第n块磁盘块的下一个块是m
    - 整个磁盘设置一张FAT表，每个盘块对应一个表目，存放连接文件各物理块的指针。

索引文件

- 为每个文件建立逻辑块号和物理块号的对照表.文件由数据文件和索引表构成。这种文件称为索引文件。 可以通过查询索引表找到每个文件的第n块逻辑块对应物理块块号
- 组织
    - 直接索引
        - 索引表就是存储数据的物理块块号
    - 一级间接索引
        - 文件目录项中有一组表项，其内容登记的是第一级索引表块的块号。第一级索引表块中的索引表项登记的是文件逻辑记录所在的磁盘块号。
    - 二级间接索引
        - 文件目录项中有一组表项，其内容登记的是第二级索引表块的块号。第二级索引表块中的索引表项登记的第一级索引表块的块号，第一级索引表项中登记的是文件逻辑记录所在的磁盘块号。

UNIX文件系统

- ① 树型文件目录结构
- ② 可安装拆卸的文件系统
- ③ 文件是无结构的字符流式文件
- ④ 将外部设备与文件一样对待
- UNIX索引表概述
    - UNIX文件系统里面磁盘块分成两个部分,一个是存放索引的索引磁盘块,一个是存放数据的磁盘块,还有一种就是存放间接索引表的
    - 索引磁盘块里面有
        - 文件所有者标识
        - 文件类型
        - 文件存取许可权
        - 地址索引表
            - UNIX第七版本

        - 小型文件
            - 小于8*磁盘块大小的文件
            - 八个索引表表项都是直接索引的
        - 大型文件
            - 大于小型文件,小于7*256*磁盘块大小的文件
            - 索引表的前7个元素都是一级间接索引
        - 巨型文件
            - 大于大型文件
            - 索引表前7个元素用于一级间接索引,最后一个用于二级间接索引
    - UNIX V
        - 索引表一共有13个表项
        - 其中前10个用于直接索引
        - 第11个用于一级间接索引
        - 第12个用于二级间接索引
        - 第13个用于三级间接索引
    - 间接索引的计算方法:磁盘块块大小/一个索引项占据的存储空间,就是一个间接索引能存放的下一级索引/数据块的个数
    - 几级间接索引代表要访问几次磁盘块才能找到数据块,一般来说n级间接索引指向n-1级间接索引
- 文件联结数目

文件目录

文件目录

- 文件目录是记录文件名字,存放地址和其他有关文件的说明信息和控制信息的数据结构.文件目录由文件目录项组成,一个文件目录项记录了一个文件的信息.
    - 一级文件目录:已建立的文件名、存放地址和有关的说明信息都放在一张表里面就称为一级文件目录
        - 特点:简单、但是检索时间长,不运行有重名的文件出现
    - 树型文件目录:目录文件就包含了这个目录下面所有数据文件和目录文件对应的文件目录项.数据文件一定在树叶上,这样一个目录是一层.
    - 文件目录就包含了这个目录里面所有文件对应的文件目录项的集合.文件目录项可以认为是文件名+FCB的集合.

目录文件

- 目录文件就是说明文件目录在系统里面是以目录文件的形式存在的,是一个具体的文件,只不过这个文件代表了目录,而不是普通的数据.

文件目录项

- 记录这个目录下面每一个文件的信息.所以说文件目录就是由若干个文件目录项组成的数组组成的.一个实例目录文件就是由若干个表示该目录下的文件的文件目录项组成的
    - 组成
        - 文件名
        - 逻辑结构
        - 物理结构
            - 如果是连续文件,就代表文件第一块物理地址和块数
            - 如果是串联文件,就代表文件第一块物理地址
            - 如果是索引文件,就代表索引表地址.
        - 其他管理信息

文件路径名

- 多级目录中，文件的路径名是由根目录到该文件的通路上所有目录文件符号名和该文件的符号名组成的字符串，相互之间用分隔符分隔。
  从根节点到这个文件经过的所有目录文件符号名

文件存储空间管理

FAT16表所有物理块的下一个块的集合

位示图

- 利用二进制的一位来表示磁盘中的一个盘块的使用情况：0表示空闲，1表示已分配。

UNIX文件存储空间的管理

- 引导块
- 管理块
    - 管理空闲磁盘块的数据结构
    - 其中有直接管理的空闲块数s_nfree
    - 还有空闲块号栈s_free
        - 成组链接法
            - 空闲磁盘块管理采用成组链接法，即将空闲表和空闲链两中方法相结合
            - 其中每一个空闲块号栈s_free里面都有100个元素,其中99个存储了空闲磁盘块的块号,最后一个s_free[0]存储的就是下一组管理块的地址,下一组管理块存储的也是100个s_free元素,也是前99个是空闲磁盘块块号,最后一个是下一个管理块,构成了一个链表结构.(前99个是实际,最后一个是下一个管理块块号.)
        - 分配算法
            - s_nfree-1
            - 如果s_nfree为0了,就把s_free[0]也就是下一个管理块载入到内存中.

        - 回收算法
            - s_nfree++;
            - 如果达到了100,就把当前管理块释放到磁盘中,然后初始化,s_free[0]为原来释放到的那个磁盘块块号.
- 索引节点(全是索引表)
- 数据区

文件的共享与安全

文件共享

- 某一个或者某一部分的文件让多个用户共同使用

文件安全

- 文件本身不得未经文件主授权的任何用户存取
- 保护方法:对用户的权限进行验证,是指用户在存取文件之前,需要检查用户的存取权限是否符合规定.

文件路径名加快文件查找

- 当前目录
    - 当前目录是当前用户正在使用的文件所在的目录。
    - 当指定当前目录后，用户对文件的所有访问都是相对于”当前目录“进行的.

链接技术

- 一个目录中的一个表目直接指向另一个目录表目对应的物理位置.一个表目 的 物理结构项就和另外一个表目的物理结构项一样

UNIX/Linux的链接

- 硬连接
    - 在索引文件中增加链接计数，用于记录共享数量。
    - 两个文件的物理结构项是一样的
- 符号链接
    - 创建一个LINK类型的新文件，文件中仅包含被链接文件的路径名

文件操作与文件备份

文件的一堆操作

- 文件的打开
    - 首先获得文件路径名
    - 按照名字查找文件目录结构获得目录项找到FCB
      注意,只用找到FCB就可以,对应的数据块不需要
    - 存入活跃文件目录表

    - 建立文件读写状态信息表,将访问指针指向文件首
- 文件关闭
    - 检查参数，获得fd；
    - 在打开文件表和文件读写状态信息表中把对应文件占用的空间释放
    - 如果“活跃文件目录表”中文件控制块不再使用，则释放该文件控制块所占的内存空间。
- 文件创建
    - 检查参数合法性
    - 建立一个文件控制块，并在目录表中建目录项。
    - 将参数填入文件控制块
    - 分配文件所存放的外存空间（也可在写数据时分配），将文件物理存储信息填入文件控制块中
- 文件删除
    - 检查参数，得到文件名（路径名）
    - 按名查找文件目录结构得到目录项，找到文件的文件控制块
    - 按文件控制块中的定位信息（如索引表）释放文件所占外存空间
    - 从文件目录结构中删除文件控制块及目录项

文件的一堆表

- 进程控制块里面有有打开文件表,记录这个进程打开了什么文件
- 对于进程的打开的许多文件有一个读写状态信息表,记录进程读或者写一个文件写到文件的哪里了
- 活跃文件目录表,就是记录所有打开过的文件的FCB,读写状态信息表中就指向活跃文件目录表

文件备份

- 周期性转储
    - 过一个周期就把存储器所有内容存一遍
- 增量性转储
    - 以文件为单位,定期转储上次转储后改过的新文件

UNIX

目录项

- 每个目录项16字节,其中1~2字节我i对应的索引节点(i节点号),剩下14字节为文件名

树型目录结构

- 每个文件系统都有一个根目录文件，它的辅存i节点是相应文件存储设备上辅存索引区中的第一个。
- 文件目录项存储的是索引节点的节点号,要获得文件,要先打开索引节点,在索引节点中根据节点地址寻找文件本身的数据.(或者是目录或者是数据)

打开文件的结构

- 活动i节点表
  - 当执行打开文件操作时，将文件辅存i节点的有关信息拷贝到主存，形成活动i节点表，他由若干个活动i节点组成。(所有打开过的文件的索引节点(i节点)都放到这个表里面)
- 系统打开文件表
  - 一个文件可以被不同进程打开,所以说构造了一个结构记录所有进程打开过什么文件
    - 文件的打开模式,读&写?
    - 引用计数:多少个进程用这个方法打开这个文件
    - 指向该文件对应的主存索引节点
- 用户文件描述符表
  - 每个进程里面的结构,用来记录这个进程用何种方式打开过何种文件,这个会指向系统打开文件表

打开磁盘块的总结

在关系里面,存在一个箭头就代表要打开箭头所指的磁盘块

n级索引表要获得磁盘块的数据要打开n+1个磁盘块

对于索引表物理结构文件,已知一个目录,打开目录里面的某个文件需要载入2个磁盘块(文件对应的索引和数据)

对于其他的,已知一个目录打开某个文件最少一个磁盘块,最多n个