操作系统

002计算机与信息技术学院.pdf

题型

第一章 计算机系统概述

操作系统概念

自上而下大致分四个部分：

- 硬件
- 操作系统：管理各种计算机硬件，为应用程序提供基础，并充当计算机硬件与用户之间的中介
- 应用程序
- 用户

操作系统的特征

- 并发
- 共享
- 虚拟
- 异步

> 操作系统最基本特征：并发、共享
>
> 两者互为存在条件

并发

两个或多个事件在同一时间间隔内发生

共享

即资源共享，指系统中的资源可供内存中多个并发执行的进程共同使用

资源共享方式：

- 互斥共享方式：在一段时间内只允许一个进程访问资源
- 同时访问方式：允许在一段时间内由多个进程“同时”访问（宏观上的同时）

虚拟

把一个物理上的实体变为若干逻辑上的对应物

虚拟处理器、虚拟存储器

操作系统的虚拟技术：

- 时分复用（处理器分时共享）
- 空分复用（虚拟存储器）

异步

进程的执行不是一贯到底的，而是走走停停的

操作系统的目标和功能

- 计算机系统资源的管理者
- 用户与计算机硬件系统之间的接口
- 实现了对计算机资源的扩充

计算机系统资源的管理者

- 处理机管理：处理机的分配和运行都以进程（或线程）为基本单位，因此也可归结为堆进程的管理
- 存储器管理：内存的分配与回收、地址映射、内存保护和内存扩充等
- 文件管理：计算机中的信息都是以文件的形式存在的，负责文件管理的部分称为文件系统
- 设备管理：完成用户的I/O请求，方便用户，提高设备利用率。包括缓冲管理、设备分配、虚拟设备等

用户与计算机硬件之间的接口

- 命令接口
    - 联机命令接口：又称交互式命令接口，适用于分时或实时系统
    - 脱机命令接口：又称批处理命令接口，适用于批处理系统

- 程序接口：由一组系统调用（也称广义指令）组成

对计算机资源的扩充

把覆盖了软件的机器称为扩充机器或虚拟机

操作系统的发展与演化

1. 手工操作阶段（此阶段无操作系统）

所有工作都要人工干预

人机矛盾越来越大（速度和资源的利用）

突出缺点：

- 用户独占全机
- CPU利用不充分

2. 批处理阶段（操作系统开始出现）

为了解决人机矛盾和CPU与IO设备之间速度的不匹配

单道批处理系统

对作业的处理是成批进行的，但内存中始终只保持一道作业

主要特征：

- 自动性：自动地逐个运行
- 顺序性：各作业顺序进入内存，先调入内存的作业先完成
- 单道性：内存中仅有一道程序

问题：

- 当运行时发出IO请求后，高速CPU就要等待低速IO

多道批处理系统

在单道批处理系统的基础上，允许多个程序同时进入内存并允许他们在CPU中交替运行，共享系统中各种软/硬件资源。当一个程序因IO请求暂停时，CPU会运行另一个程序。

特点：

- 多道：内存中同时存放多道独立的程序
- 宏观上并行
- 微观上串行

产生的问题：

- 处理及分配
- 内存分配
- IO设备分配
- 如何保证安全性与一致性

优点：

- 资源利用率高
- 吞吐量大

缺点：

- 用户响应时间长
- 不提供人机交互功能

分时操作系统

把处理器的运行时间分成很短的时间片，按时间片轮流分配处理器

主要特征：

- 同时性：运行多个终端用户同时使用一台计算机
- 交互性：很好的解决了人机交互问题
- 独立性：多个用户可以彼此独立地进行操作
- 及时性：用户请求能在短时间内被响应

实时操作系统

- 硬实时系统：某个动作必须绝对地在规定时间时刻（或规定时间范围）发生
- 软实时系统：能接受偶尔违反时间规定且不会引起任何永久性的损害

主要特点：

- 及时性
- 可靠性

操作系统运行环境

处理器运行模式

CPU执行两种不同性质的程序：

- 操作系统内核程序
- 用户自编程序

两种指令：

- 特权指令：不允许用户直接使用的指令
- 非特权指令：允许用户直接使用的指令，仅限于访问用户的地址空间

CPU的运行模式划分为：

- 用户态（目态）
- 核心态（管态、内核态）

大多数内核包括4方面的内容：

- 时钟管理
- 中断机制
- 原语
- 系统控制的数据结构及处理

时钟管理

功能：

- 计时
- 通过时钟终端的管理，实现进程切换

中断机制

中断机制中只有一小部分功能属于内核，他们负责保护和回复中断现场的信息；转移控制权到相关的处理程序

原语

特点：

- 处于操作系统最底层，最接近硬件

- 有原子性
- 运行时间短，调用较频繁

系统控制的数据结构及处理

进程控制块、文件控制块等

中断和异常

中断：也称外中断，指来自CPU执行指令外部的事件（如输入输出）

异常：也称内中断，指来自CPU执行指令内部的事件（如非法操作符、越界等）

> 异常不能被屏蔽

中断和异常的分类

- 中断
  - 内中断
    - 故障：指令执行引发的异常
    - 自陷：用于在用户态下调用内核程序
    - 终止：出现了导致CPU无法运行的硬件故障
  - 外中断（硬件）
    - 可屏蔽中断：可通过屏蔽字实现多重中断
    - 不可屏蔽中断：紧急硬件故障，如电源掉电

中断和异常的处理过程

CPU打断当前用户程序，转到相应的中断或异常处理程序（具体过程由操作系统完成）

发现不可恢复的致命错误，终止用户程序

系统调用

指用户在程序中调用操作系统所提供的一些子功能（特殊的公共子程序）

在用户程序中，凡是与资源有关的操作都必须通过系统调用的方式向操作系统提出请求，并由操作系统代为完成

系统调用大致可分为如下五类：

- 设备管理
- 文件管理
- 进程控制
- 进程通信
- 内存管理

系统调用的过程：

1. 传参
2. 陷入指令/访管指令（抛出自陷异常）
3. 由操作系统内核处理系统处理请求
4. 返回应用程序

操作系统结构

分层法

将操作系统分为若干层

最高层为用户结构，最底层（0层）为硬件

每层都只能调紧邻它的最低层（单向依赖）

优点：

- 便于调整和验证，方便设计和实现
- 易扩充和维护

缺点：

- 合理分层较难
- 效率差

模块化

将操作系统按功能划分成若干具有一定独立性的模块

模块中可进一步分子模块

衡量模块独立性的标准：

- 内聚性：模块内部各部分之间连接的紧密程度，内举行越高独立性越好

- 耦合性：模块间相互影响的程度，耦合性越高独立性越差

优点：

- 正确性、可理解性和可维护性
- 可适应性
- 开发快速

缺点：

- 模块间的接口难以设计合理
- 模块间相互依赖，更难调试和验证

宏内核

从内核架构来分，可分宏内核和微内核

宏内核：也称大内核或单内核，指操作系统将主要功能模块都作为一个紧密连接的整体运行在核心态

优点：高性能

缺点：代码量大、结构混乱、难以维护

微内核

基本概念

又称小内核

将最基本的功能保留在内核，不需要再核心态执行的功能移到用户态执行

微内核功能

”机制和策略分离“：机制部分放入微内核

- 进程（线程）管理
- 低级存储器管理
- 中断和陷入处理

微内核特点

优点：

- 扩展性和灵活性

- 可靠性和安全性
- 可移植性
- 分布式计算

缺点：

- 需要频繁地再核心态和用户态切换，性能低

外核

在底层有外核程序在内核态运行

内核负责进程调度、进程通信等功能

外核负责为用户分配未经抽象的硬件资源，且由外核保证其安全性

优点：

- 外核可以分配不抽象、不虚拟的资源，更加灵活
- 减少虚拟，提升效率

缺点：

- 降低系统一致性
- 系统更加复杂

操作系统引导

引导过程：

1. 激活CPU：激活CPU读取ROM中的boot程序，即开始执行BIOS的指令
2. 硬件自检
3. 加载带有操作系统的硬盘：BIOS开始读取Boot Sequence（通过CMOS里保存的启动顺序，或交互的方式），把控制权交给启动顺序排在第一位的存储设备
4. 加载主引导记录MBR：磁盘以特定的标识符区分引导硬盘和非引导硬盘，主引导记录就是告诉CPU去硬盘中哪个主分区（含有操作系统的分区，又称活动分区）找操作系统
5. 扫描硬盘分区表，并加载硬盘活动分区：将活动权交给活动分区
6. 加载分区引导记录PBR：读取活动分区的第一个扇区（称为分区引导记录PBR），寻找并激活分区根目录下用于引导操作系统的程序（启动管理器）
7. 加载启动管理器：在PBR中寻找启动管理器，加载
8. 加载操作系统

第二章 进程与线程（处理机管理+同步通信及死锁）

进程与线程

进程

典型定义：

- 是程序的一次执行过程
- 是一个程序及其数据在处理及上顺序执行时发生的活动
- 进程是具有独立功能的程序在一个数据集合上运行的过程

进程是系统进行资源分配和调度的一个独立单位

进程控制块PCB：利用PCB来描述过程的基本情况和运行状态，是进程存在的唯一标志

进程实体（又称进程映像）由三部分组成：

- 程序段
- 相关数据段
- PCB

> 进程是动态的，进程映像是静态的

在引入线程之前，进程是调度的基本单位

进程的特征

特征也是基本要求：

- 动态性：最基本特征
- 并发性
- 独立性
- 异步性

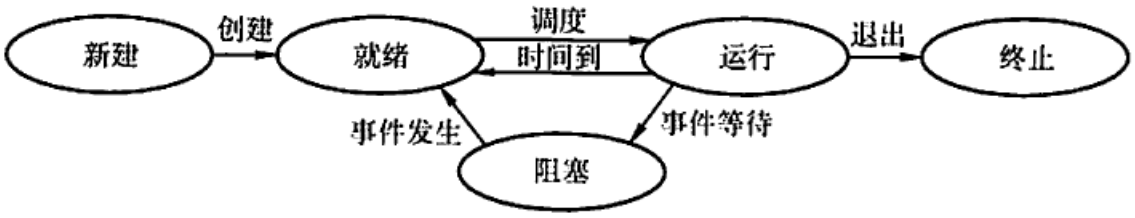
进程的状态与转换

进程状态：

- 运行态：正在处理机上运行
- 就绪态：进程获得了除处理机之外的一切所需资源，就绪队列

- 阻塞态：又称等待态，正在等待某一事件而暂停运行
- 创建态：正在被创建，尚未到达就绪态
- 结束态：进程正在从系统中消失

> 运行态、就绪态、阻塞态称为3个基本状态



进程的组织

进程由以下三个部分组成：

- 进程控制块PCB
- 程序段：调度到CPU执行的程序代码段，程序可被多个进程共享
- 数据段

进程控制

原语：执行期间不允许中断，是不可分割的基本单元，通过开/关中断实现，是一种特殊的程序

进程的创建

父进程

子进程

系统创建一个新进程的过程如下（创建原语）：

1. 为新进程分配一个进程标识号；
2. 为进程分配其运行的所需资源，如果所需资源不足，则保持创建态，等待资源；
3. 初始化PCB；
4. 若进程能够进入就绪队列，则插入就绪队列；

进程的终止

引起进程终止的事件：

- 正常结束
- 异常结束
- 外界干扰

系统终止进程的过程如下（终止原语）：

1. 根据标识符检索出PCB，读出进程状态
2. 若被终止进程处于执行状态，则结束执行，将处理器分配给其他进程
3. 若该进程有子进程，则结束子进程
4. 释放、归还所有资源
5. 将PCB从所在队列中删除

进程的阻塞和唤醒

进程通过调用阻塞原语，使自己由运行态变为阻塞态（自发的）

系统阻塞进程的过程如下（阻塞原语）：

1. 根据标识符检索出PCB
2. 若进程为运行态，则保护现场，将其转化为阻塞态，停止运行
3. 把PCB插入相应事件的等待队列，将处理机资源调度给其他就绪进程

当阻塞进程期待的事件发生时，由有关进程调用唤醒原语，唤醒等待该事件的进程

系统唤醒阻塞进程的过程如下（唤醒原语）：

1. 从该事件的等待队列找到PCB
2. 将其从等待队列中移出，转化为就绪态
3. 把PCB插入就绪队列

阻塞原语和唤醒原语作用正好相反，必须成对使用。

进程的通信

进程通信：进程之间的信息交换

低级通信方式：PV操作

高级通信方式：

- 共享存储
- 消息传递
- 管道通信

共享存储

在通信的进程之间存在一块可直接访问的共享空间，通过对共享空间进行读写操作实现进程的之间信息交换

需要同步互斥工具（PV操作）对共享空间进行读写控制

共享存储又分两种：

- 低级：基于数据结构的共享
- 高级：基于存储区的共享

操作系统只负责提供共享的存储空间和同步互斥工具，数据交换由用户自己安排读写执行实现。

消息传递

进程间的数据交换以格式化的消息为单位

系统提供发送消息和接受消息两个原语进行数据交换

最广泛的通信机制

消息传递又分：

- 直接通信方式：直接发送给接收进程，挂载到接收的消息缓冲队列上
- 间接通信方式：发送给中间实体（信箱），接收进程从中间实体取得消息

管道通信

管道：用于连接一个读进程和写进程以实现通信的一个共享文件（通道文件）

读进程和写进程都以字符流进行读写

先进先出

一个管道只能单向通信

读数据是一次性操作，即读一次后，共享文件中相应内容就被移除了

管道机制必须提供三个协调能力：

- 互斥
- 同步
- 确定对方存在

> 真题：一个管道允许多个写进程，一个读进程
>
> linux：允许多个进程读写

线程和多线程模型

线程的基本概念

一个基本的CPU执行单元

程序执行的最小单元

线程可以与同一个进程的所有线程共享进程拥有的全部资源

一个线程可以创建和撤销另一个进程，同一个进程的多个线程可并发

线程也有就绪、阻塞和运行三种状态

线程控制块TCB

进程与线程的比较

	线程	进程
调度	独立调度的基本单位 线程切换不一定会引起进程切换	每次调度都要进程上下文切换，开销大；
并发性	可并发	可并发
拥有资源	不拥有资源（仅有一些必不可少的、保证独立运行的资源）	拥有资源的基本单位
独立性	共享进程的地址空间和资源	不同进程之间仅共享全局变量，拥有独立的地址空间和资源
系统开销	切换时只保存少量寄存器内容，开销小	
支持多处理机系统	可将进程中的多个线程分配到多个处理机上执行	不支持线程时只支持在一个处理机上运行

线程的实现方式

分两类：

- 用户级线程
- 内核级线程：又称内核支持的线程

用户级线程ULT

线程的管理（创建、撤销和切换）的所有工作都由应用程序在用户空间中完成，内核意识不到用户级线程的存在

优点：

- 线程切换不需要转换到内核态，节省模式切换的开销
- 不同进程可根据自身需要，对自己的线程选择不同调度算法
- 线程的实现与操作系统无关

缺点：

- 系统调用的阻塞问题，当一个线程被阻塞，同一进程的所有线程都会被阻塞
- 不能发挥多处理机的优势

内核级线程KLT

在内核的支持下运行，线程管理是在内核态下实现的

优点：

- 能发挥多处理机的优势
- 如果一个线程被阻塞，同一进程的其他线程可继续调度运行
- 内核支持线程具有较小的数据结构和堆栈，切换块、开销小
- 内核本身也可以采用多线程技术，提供系统的执行效率和速度

缺点：

- 同一进程的线程切换时需要转化到核心态，系统开销大

组合方式

内核支持多个内核级线程，同时允许用户程序支持用户级线程

多线程模型

在同时支持内核级线程和用户级线程的系统中，有以下面三种不同的多线程模型：

- 多对一模型：多个用户级线程映射到一个内核级线程，这些线程属于同一个进程

  优点：线程管理效率高

  缺点：一个线程被阻塞，则整个进程都会被阻塞，并发度不高

- 一对一模型：每个用户级线程对应一个内核级线程

优点：当一个线程被阻塞，不影响同进程的其他线程，并发度高

缺点：每创建一个用户线程，就要创建一个相应的内核线程，开销大

- 多对多模型：n个用户线程映射到m个内核级线程

  特点：克服了前两个模型的缺点，兼有前两个模型的优点

调度的概念

多道系统中，进程的数量往往远多于处理机的个数

处理机调度就是对处理机进行分配

处理机调度是多带程序操作系统的基础

调度的层次

- 高级调度（作业调度）：内存与辅存之间的调度

  从外存上处于后备队列的作业中挑选一个或多个，给他们分配内存、设备等必要资源，并建立相应进程

  每个作业只调入一次，调出一次

  主要是多道批处理系统中配有作业调度

- 中级调度（内存调度）：将暂时不能运行的进程调度到外存中等待（挂起态）

  当挂起态的进程具备运行条件且内存有空闲时，调入内存（挂起态->就绪态）

- 低级调度（进程调度）：为就绪队列中的进程分配处理机

  最基本的调度

作业调度为进程的活动做准备，进程调度使进程真正的运行起来，内存调度将暂时不能运行的进程挂起

调度的目标

由很多评价标准：

- CPU利用率：CPU有效工作时间/总时间
- 系统吞吐量：单位时间CPU完成作业的数量
- 周转时间：作业完成时间–作业提交时间
  - 平均周转时间
  - 带权周转时间：周转时间/实际运行时间

- 等待时间：处于等待处理机的时间之和
- 响应时间：首次提交请求到首次产生响应的时间

调度的实现

临界区和临界资源

临界资源：一段时间内只允许一个进程使用的资源（互斥访问）

临界区：访问临界资源的代码段

内核程序临界区：一般是用来访问某种内核数据结构的（如就绪队列）

调度程序（调度器）

操作系统中用于调度和分配CPU的组件，通常由以下三部分组成：

- 排队器
- 分派器
- 上下文切换器

调度的时机、切换与过程

调度的时机：

- 创建新进程
- 进程退出
- 进程阻塞
- 中断
- 对于抢占式，每个时钟中断或k个时钟中断都会触发

> 对于支持内核级现成的系统，调度的对象是线程

不能进行调度的情况：

- 处理中断时
- 进程在操作系统的内核临界区时
- 原子操作过程中

进程调度方式

通常有以下两种：

- 非抢占式（非剥夺）：直到进程运行完成或进入阻塞态时，才把处理机分配给其他进程

  不能用于分时系统和实时系统
- 抢占式（非抢占）：允许按照某种原则取暂停正在执行的进程

  优先权、短进程优先、时间片

闲逛进程

如果系统中没有就绪进程，就会调用闲逛进程（有限度最低）

闲逛进程不需要除CPU外的任何资源（不会被阻塞）

典型的调度算法

先来先服务（FCFS）

不会导致饥饿

最简单

非剥夺

可用于作业调度和进程调度

每次从就绪队列中选择最先进入该队列的进程

特点：

- 简单，但效率低
- 长作业有利
- 有利于CPU繁忙型作业，不利于IO繁忙型作业

短作业优先（SJF）

默认为非剥夺

对短作业（进程）优先调度

特点：

- 长作业不利，“饥饿”现象
- 未考虑作业的紧迫程度
- 长短根据用户提供的估计时间确定

- 在所有进程同时到达的情况下，SJF的平均等待时间和平均周转时间最少

优先级调度