chapter 2 进程的控制与描述

Created	@October 21, 2021 2:43 PM
Tags	

前驱图和程序执行

前驱图

> 有向无循环图，可记为DAG（directed acyclic graph），用于描述程序之间执行的先后顺序

程序的顺序执行

特征

1. 顺序性
2. 封闭性
3. 可再现性

程序的并发执行

特征

1. 间断性
2. 失去封闭性
3. 不可再现性

进程的描述

PCB

进程控制块就是存放进程管理和控制信息的数据结构，是进程存在的唯一标志

> PCB内包含的信息体现了操作系统的进程的管理，如为了**区分进程**，需要记录PID，UID；为了**管理资源**，需要记录分配的资源和正在使用的文件等等；为了记录进程的运行情况**对进程进行控制和调度**，需要记录CPU使用时间，磁盘使用情况，网络流量等等

PCB中包含的信息

1. 进程标识符
    a. 外部标识符
    b. 内部标识符
2. 处理机状态
3. 进程调度信息
    a. 进程状态
    b. 进程优先级
    c. 进程调度所需的其他信息
    d. 事件（阻塞原因）
4. 进程控制信息
    a. 程序和数据的地址
    b. 进程同步和通信机制
    c. 资源清单
    d. 链接指针

> PCB的作用是使一个在多道程序环境不能独立运行的程序（含数据）成为一个能独立运行的基本单位，一个能与其他进程并发执行的进程

1. 作为独立运行基本单位的标志
2. 能实现间断性运行方式
3. 提供进程管理所需要的信息
4. 提供进程调度所需要的信息
5. 实现与其他进程的同步和通信

进程的定义

进程的引入是为了能使程序并发执行，并且可以对程序加以描述和控制

进程实体

> 进程控制块（PCB—process control block），程序段，相关数据段和PCB构成了进程实体，简称为进程，创建进程实质上是创建进程实体中的PCB，撤销也是撤销PCB

一种**进程的定义**：

> **进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位**

进程是**动态的**，进程实体是**静态的**

进程的特征：

1. 动态性：动态产生消亡
2. 并发性
3. 独立性：进程是能独立运行，独立获得资源，独立接受调度的基本单位
4. 异步性：需要进程同步机制来解决异步问题

进程与程序区别

1. 程序是静态的，进程是动态的。程序存放在磁盘的可执行文件，是一系列指令集合；进程是程序的一次执行过程
2. 进程是暂时的，程序是永久的。进程是一个状态变化的过程，程序可长久保存
3. 进程与程序的组成不同。进程的组成包括程序，数据和进程控制块
4. 进程与程序的对应关系。通过多次执行，一个程序可对应多个进程，通过调用关系，一个进程可包括多个程序

进程的基本状态及转换

基本状态：

1. ready（可放于外存（挂起））

> 即进程已处于准备好运行的状态，分配到除CPU以外的所有资源，只要获得CPU即可执行

2. running

> 获得CPU处于执行状态，单处理机系统只有一个进程处于执行状态，多处理机系统有多个处于执行状态

3. block（可放于外存（挂起））

> 正在执行的进程由于发生某个事件而暂时无法执行的状态，进程的执行受到阻塞。此时会引起进程的调度，处于阻塞状态的进程会排成一个阻塞队列，根据阻塞原因不同，会设置多个阻塞队列

4. 创建

> 进程正在被创建，为进程分配资源，初始化PCB

5. 终止

> 回收CPU使用，回收资源和PCB

基本状态的转换

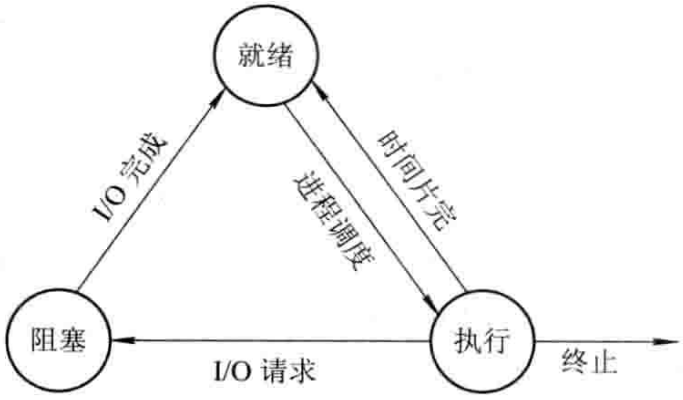


图 2-5　进程的三种基本状态及其转换

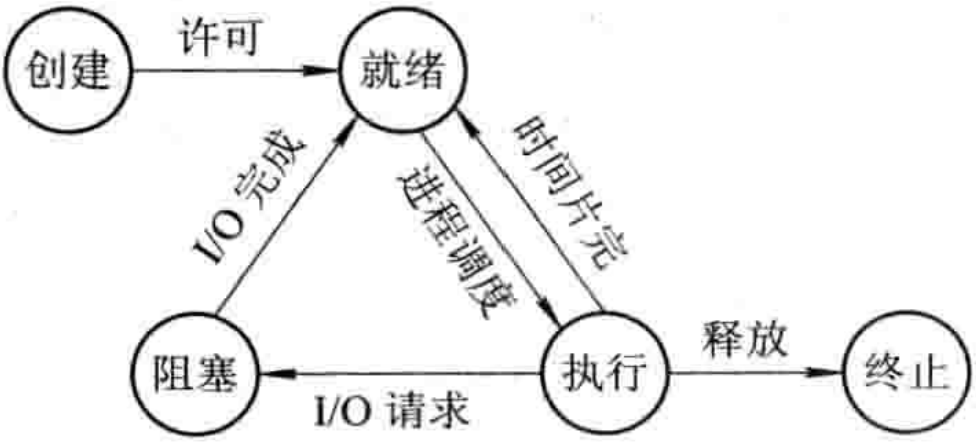


图 2-6　进程的五种基本状态及转换

挂起操作↔激活操作

引起挂起的原因：

1．终端用户的请求

2．父进程的请求

3．负荷调节的需要

4．操作系统的需要

引入挂起（suspend）之三种进程状态的转换：

1．活动就绪→静止就绪

2．活动阻塞→静止阻塞

3．静止就绪→活动就绪（激活）

4．静止阻塞→活动阻塞（激活）

引入挂起操作之后的五个进程状态转换

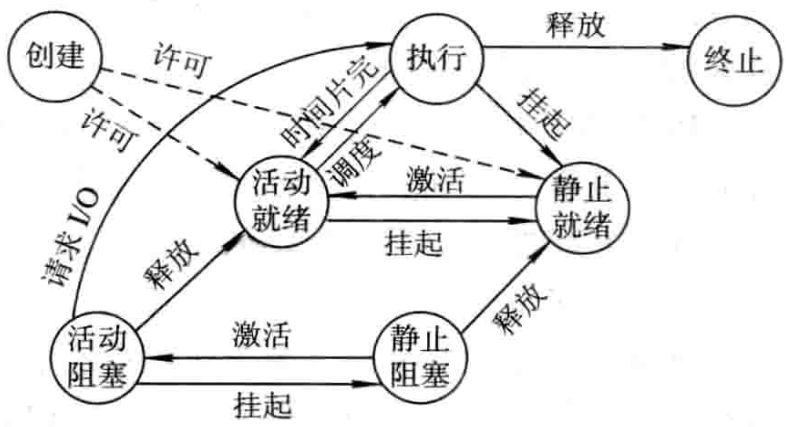
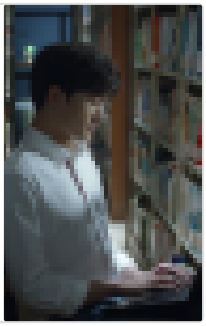


图 2-8　具有创建、终止和挂起状态的进程状态图

创建→静止就绪：当前系统的资源和性能不分配给新建进程资源，主要是主存，进程状态转为静止就绪，被放置在外存不参与调度

> 更多有关挂起的资料
>
> 操作系统--CPU和内存、挂起和阻塞_My Blogs-CSDN博客
> CPU ≠ 内存 不妨把外存比作一个大仓库，里面有各种原材料，可以生产不同的东西。而CPU就是进行加工处理的车间。 一般来说，一个时间段，想要生产的东西也就那几样（是确定的，也是有限
> https://blog.csdn.net/weixin_37641832/article/details/83217104



进程管理中的数据结构

类型：

1. 内存表
2. 设备表
3. 文件表
4. PCB

进程控制块的组织方式

1. 线性方式
2. 链接方式

> 把具有相同状态进程的PCB分别通过PCB中的链接字链接成一个队列.就绪队列和若干阻塞队列

3．索引方式

> 根据所有进程状态的不同创建几张索引表

进程控制

> **进程控制是进程管理功能中最基本的功能(控制，同步，通信，调度）**，主要包括创建新进程，终止已完成的进程，将发生异常的进程置于阻塞，负责进程运行过程中的状态转换等功能

控制办法

使用**原语**

内核

> 一些与硬件紧密相关的模块，各种常用设备的驱动程序以及运行频率较高的模块都安排在紧靠硬件的软件层次，常驻内存，通常被称为OS内核

目的

1．对软件进行保护

2．提高OS效率

同时将处理机的状态分为

1．系统态（内核态）

> 访问所有的寄存器和存储区

2．用户态（目态）

> 访问指定的寄存器和存储区

内核的功能

1. 支撑功能
    a. 中断处理
        > 内核最基本功能，操作系统活动的基础
    b. 时钟管理
    c. 原语操作
2. 资源管理功能
    a. 进程管理
    b. 存储器管理
    c. 设备管理

1. 进程的创建

进程的层次结构

UNIX中进程与其子孙进程共同组成一个进程家族

Windows中不存在进程层次结构，变成获得句柄，控制与被控制的简单关系

进程图

引起创建进程的事件

1. 用户登陆
2. 作业调度
3. 提供服务
4. 应用请求

进程的创建过程

申请PCB→为新进程分配资源→初始化进程控制块PCB→如果就绪队列接纳，放入就绪队列

2. 进程的终止

引起终止事件：

1. 正常结束
2. 异常结束
    a. 越界错
    b. 非法指令
    c. 等待超时
    d. 运行超时
    e. 算术错误
3. 外界干预

进程的终止过程

> 找出被终止进程PCB→若为运行态，置CPU调度标志为真，表示进程终止→若有子孙进程，终止子孙进程并回收资源→回收被终止进程的资源→回收被终止进程的PCB

3. 进程的阻塞与唤醒

引起阻塞事件：

1. 向系统请求共享资源失败
2. 等待某种操作的完成
3. 新数据尚未达到
4. 等待新任务到达

`block`

`wakeup`

4. 进程的挂起与激活

`suspend`

`active`

进程同步

⛵ 进程同步相关

经典进程控制问题

生产者消费者问题

> 同步和互斥同时存在

哲学家进餐

> 只存在互斥

三种解决办法：

1. 至多允许四位哲学家同时去拿左边的筷子
2. 仅当哲学家左右两边的筷子均可用时才允许拿起筷子
3. 奇数号哲学家先拿起左边的筷子，再拿起右边的筷子。偶数号哲学家相反

读写者问题

> 读写互斥，读读不互斥，写写互斥

设置一个count变量来记录当前有多少个正在读的进程

管程

> 管程的引入是为了解决信号量机制编程麻烦，易出错等问题，管程本质上是一个类（class）

组成

1. 共享的数据结构
2. 对数据结构初始化的语句
3. 访问数据结构的函数

基本特征

1. 各外部进程线程只能通过管程提供的特定入口才能访问共享数据
2. **每次仅允许一个进程在管程内执行某个内部过程**
3. 进程互斥访问管程是由**编译器**实现的

4．可在管程中设置条件变量，等待唤醒操作以解决同步问题

进程通信

> 进程间交换信息

低级通信

> 互斥和同步（信号量机制）

高级通信

> 共享存储器，消息传递系统，管道通信系统，客户机服务器

1．**共享存储**

   a．基于数据结构（低级）

   b．基于存储区（高级）

2．**管道通信**

> 固定大小的缓冲区，一个管道只能**半双工通信**，两个可一起实现双向同时通信，进程互斥访问管道
>
> > **写满才读，读完才写，读完数据被抛弃只能有一个读进程**

3．**消息传递**

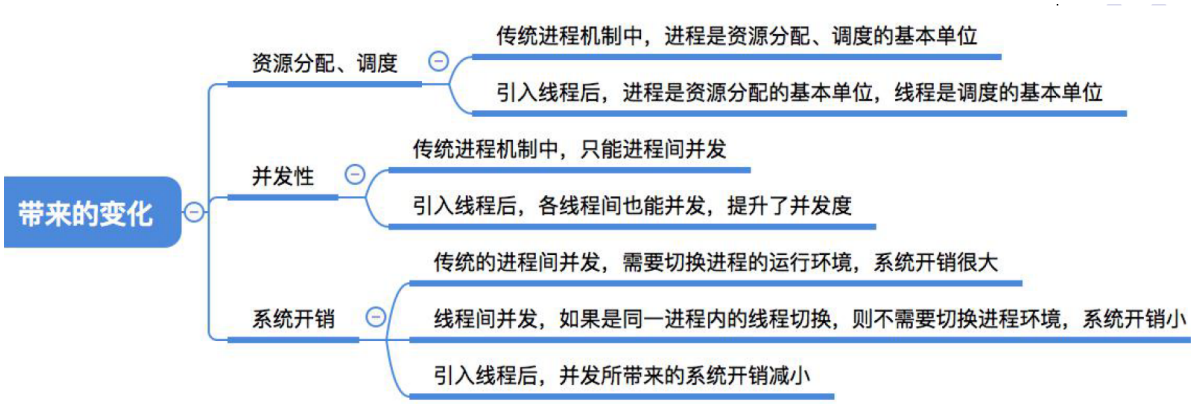
> 格式化消息为单位（消息头，消息体）类似于报文（计算机网络）
>
> 发送接收原语
>
>   a．直接通信方式：消息缓冲队列 `send receive`
>
>   b．间接通信方式

> 中间体（信箱）

线程

> 线程的引入提高了系统的并发度

引入线程后，进程只作为除CPU之外的系统资源的分配单元，线程是最基本的CPU执行单元，也是程序执行流的最小单位，线程之间可以并发，是处理机的分配单元。



线程的属性

> 线程是处理机调度的单位
>
> 多CPU，各个线程可以占用不同的CPU
>
> 线程有线程ID，线程控制块（TCB）
>
> 线程有就绪，阻塞，运行三种基本状态
>
> 线程几乎不拥有系统资源
>
> 同一进程的不同线程之间共享进程的资源
>
> 共享内存地址空间，线程通信无需系统干预
>
> 同进程线程切换不会引起进程切换

线程的实现

- 用户级线程

> 通过线程库实现对线程的管理工作，管理工作由应用程序负责，在用户态下即可完成线程切换，用户级线程对操作系统而言是透明的
>
> 优点：在用户态即可完成线程切换，开销小，效率高
>
> 缺点：一个用户级线程被阻塞之后整个进程都会被阻塞，并发度不高

- 内核级线程

> 内核级线程的管理工作由内核完成（包括线程调度和切换）
>
> 操作系统会为内核级线程建立相应的TCB
>
> 优点：当一个线程被阻塞之后，其他线程可以继续执行，并发性好
>
> 缺点：一个用户进程会占用多个内核级线程，线程切换在核心态，成本高开销大

多线程模型

1．一对一模型

> 一个用户级线程映射一个内核级线程

2．多对一模型

> 多个用户级线程映射一个内核级线程
>
> **内核级线程才是处理机分配的基本单位**

3．多对多模型