# 进程管理

## 3.1 进程的概念

### 3.1.1 程序的并发执行

**1. 程序**

程序描述计算机所要完成的具有独立功能，并在时间上按严格次序前后相继的计算机操作序列集合，是一个静态的概念。

1. **程序的顺序执行**

计算机的CPU是通过时序脉冲来控制顺序执行命令的。我们把一个具有独立功能的程序独占处理机直至最终结束的过程称为程序的顺序执行。有以下特点：

1. 顺序性

每执行一条指令，系统将从上一个执行状态转移到下一个执行状态，且上一个指令的执行结束是下一条指令执行开始的充分必要条件。

1. 封闭性

程序执行得到的最终结果由给定的初始条件决定，不受外界因素影响。

1. 可再现性

与执行速度无关，只要输入的初始条件相同，则无论何时重复该程序都会得到相同的结果。

1. **多道程序系统中程序执行环境的变化**
2. 独立性
3. 随机性
4. 资源共享性
5. **程序的并发执行**
6. 什么是程序的并发执行

所谓并发执行，是为了增强计算机系统处理能力和提高资源利用率所采用的一种同时操作技术。并发执行分为两种：

①多道程序系统的程序执行环境变化所引起的多道程序的并发执行

②在某道程序的几个程序段中包含着一部分可以同时执行或顺序颠倒执行的代码

总结：一组在逻辑上相互独立的程序或程序段在执行过程中，其执行时间在客观上相互重叠，即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的这种执行方式。

两个相邻语句S1,S2可以并发执行的条件：

R（S1） ∩ W（S2）= {空集}

W（S1） ∩ R（S2）= {空集}

W（S1） ∩ W（S2）= {空集}

1. 程序的并发执行所带来的影响

程序的并发执行充分地利用了系统资源，从而提高了系统地处理能力，这是并发执行好地一面。但是由于资源有限，程序的并发执行必然导致资源共享和竞争。因此需要有一个能描述程序的执行过程且能用来共享资源的基本单位，这个基本单位被称为进程（或任务）。

### 3.1.2 进程的定义

定义：并发执行的程序在执行过程中分配和管理资源的基本单位。

进程和程序的区别和联系：

1. 进程是动态概念，程序是静态概念。
2. 进程具有并发特征，而程序没有。（程序是静态的，不反应执行过程
3. 进程是竞争计算机系统资源的基本单位，从而其并发性受到系统自己的制约。
4. 不同进程可以包含同一程序，只要该程序所对应的数据集不同。

## 3.2 进程的描述

进程的静态描述由3部分组成：进程控制块（PCB），有关程序块和该程序段对其进行操作的数据结构集。

### 3.2.1 进程控制块

1. 描述信息

2. 控制信息

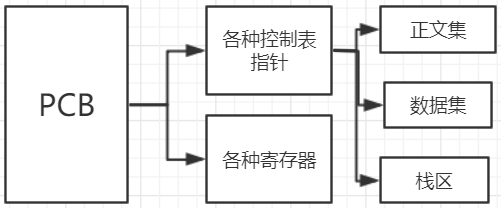
3. 资源管理信息

4. CPU现场保护结构

### 3.2.2 进程上下文

进程上下文实际上是进程执行过程中顺序关联的静态描述。进程上下文是一个与进程切换和处理机状态发生交换有关的概念。

已执行过的进程指令和数据在相关寄存器与堆栈中的内容称为上文，正在执行的指令和数据在相关寄存器与堆栈中的内容称为正文，待执行的进程指令和数据在相关寄存器与堆栈中的内容称为下文。



### 3.2.3 进程上下文切换

进程上下文切换一般包括3个部分，并涉及3个进程。

第一部分是保存被切换进程的正文部分（或当前状态）至有关存储区，例如PCB。

第二部分是操作系统进程中有关调度和资源分配程序执行，并选取新的进程。

第三部分是将被选中进程的原来被保存的正文部分从有关存储区中取出，并送至有关寄存器和堆栈中，激活被选中进程执行。

### 3.2.4 进程空间与大小

任一进程，都有一个自己的地址空间，该空间称为进程空间或虚空间。

进程空间的大小只与处理机的位数有关。

## 3.3 进程状态及其转换

### 3.3.1 进程状态

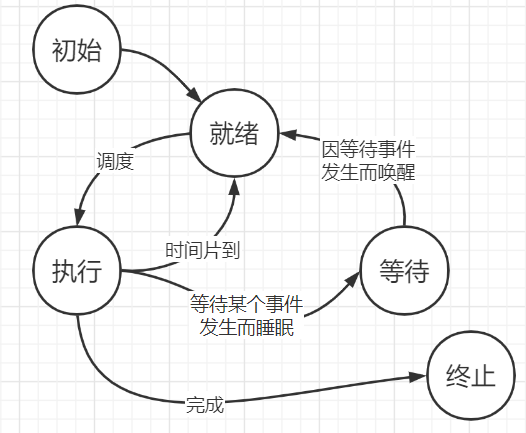
在进程的生命，一个进程至少具有5种基本状态：初始态，执行状态，等待状态，就绪状态和终止状态。

处于就绪状态的进程已经得到除CPU之外的其他资源，只要由调度得到CPU，便立即投入执行。就绪状态又可进一步分为内存就绪状态和外存就绪状态；只有成为内存就绪状态后，才可以被执行。

在单CPU系统中，任意时刻处于执行状态的进程只能有一个。

进程的执行状态由于用户程序和操作系统内核程序又分为用户态和系统态。

### 3.3.2 进程状态转换



## 3.4 进程控制

进程和处理机的一个重要任务就是进程控制。所谓进程控制，就是系统使用一些具有特定功能的程序块来创建，撤销进程以及完成进程各状态间的转换，从而达到多进程高效率并发执行和协调，实现资源共享的目的。

一般地，把系统态下执行的某些具有特定功能的程序段称为原语。

原语可分为两类：

1.机器指令级的，特点是执行期间不允许中断。

2.功能级的，其特点是作为原语的程序段不允许并发执行

### 3.4.1 进程创建与撤销

**1. 进程创建**

进程创建的方式：

1. 由系统程序模块统一创建

例如：在批处理系统中，由操作系统的作业调度程序为用户作业创建相应的进程以完成用户作业所要求的功能。

由系统统一创建的进程之间的关系是平等的，它们之间一般不存在资源继承关系。

1. 由父进程创建

例如：在层次结构的系统中，父进程创建子进程以完成并行工作。

父进程与父进程创建的进程之间则存在隶属关系，且互为树形结构的家族关系。

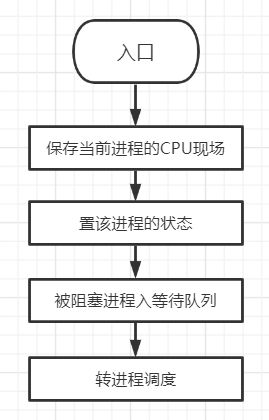
1. **进程撤销**

以下几种情况导致进程被撤销：

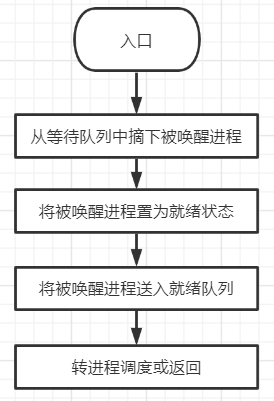
1. 该进程已完成所要求的功能而正常终止（寿终正寝
2. 由于某种错误导致非正常终止（半途而废
3. 祖先进程要求撤销某个子进程（爸爸打儿子

### 3.4.2 进程的阻塞与唤醒

阻塞：进程由执行状态到等待状态



唤醒：进程由等待状态到执行状态



## 3.5 进程互斥

### 3.5.1 资源共享所引起的制约

**1.临界区**

把不允许多个并发程序交叉执行的一段程序称为临界部分或临界区

临界区是由属于不同并发进程的程序段共享公用数据或公用变量而引起的

**2.间接制约**

这种由于共享某一公有资源而引起的在临界区内不允许并发进程交叉执行的现象，称为由共享公有资源而造成的对并发进程执行速度的间接制约，简称间接制约。

**3.什么是互斥**

互斥的定义：一组并发程序中的一个或多个程序段，因共享某一公有资源导致它们必须以一个不允许交叉执行的单位执行；简单来说就是多个并发进程只有一个可以进入临界区。

一组并发进程互斥执行时必须满足如下准则：

1. 不能假设各并发进程的相对执行速度。即各并发进程享用平等地，独立地竞争共有资源的权利，且不采取任何措施的条件下，在临界区内任一指令结束时，其他并发进程可以进入临界区。
2. 并发进程中的某个进程不在临界区时，它不阻止其他进程进入临界区。
3. 并发进程的若干个进程申请进入临界区时，只能允许一个进程进入。
4. 并发进程中的某个进程从申请进入临界区时开始，应在有限时间内得以进入临界区。

（1）（2）（3）是保证并发进程享有平等地，独立地竞争和使用公有资源的权利，且保证每一时刻至多只有一个进程在临界区。而准则（4）则是并发进程不发生死锁的重要保证。

### 3.5.2 互斥的加锁实现

1. 对临界区加锁实现互斥

### 3.5.3 信号量和P、V原语

**1. 信号量** （看门的老大爷

信号量管理相应临界区的公有资源，它代表可用资源实体。

在操作系统中sem是一个整数。在sem大于等于零时代表可供并发进程使用的资源实体数，但sem小于零时则表示正在等待使用临界区的进程数。

**2.P、V原语**

信号量的数值仅能由P，V原语操作改变

## 3.6 进程同步

### 3.6.1 同步的概念

把异步环境下的一组并发进程因直接制约而互相发送消息而进行互相合作，互相等待，使得各进程按一定的速度执行的过程称为同步。

### 3.6.2 私用信号量

一般来讲可以把各进程之间发送的消息作为信号量看待。但是与互斥不同的是，这里的信号量只与制约进程及被制约进程有关而不是与整组并发进程有关。因此，称该信号量作为私用信号量。

### 3.6.3 用P、V原语操作实现同步

## 3.7 进程通信

通信意味着在进程间传送数据。

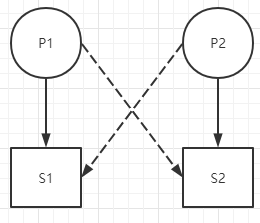
一般来说，进程间的通信根据通信内容可以划分为两种，即控制信息的传送（低级通信）与大批量数据传送（高级通信）。

## 3.8 死锁问题

### 3.8.1 死锁的概念

**1.死锁的定义**

定义：各并发进程互相等待双方所拥有的资源，且这些并发进程在得到对方资源之前不会释放自己所拥有的资源。从而造成大家都想得到资源而又都得不到资源，各并发进程不能继续向前推进的状态。

虚线代表需求；实线代表拥有

**2.死锁的起因**

死锁的起因是并发进程的资源竞争，根本原因在于系统提供的资源个数少于并发进程所要求的该类资源数。

**3.产生死锁的必要条件**

（1）互斥条件。

（2）不剥夺条件。进程所获资源在未使用完毕之前，不能被其他资源强行剥夺，只能自己释放，

（3）部分分配。每次申请部分资源，在等待新资源的同时，继续占用已分配的资源

（4）环路条件。存在一种进程循环链，链中每一个进程已获得的资源同时被下一个进程所请求。

### 3.8.2 死锁的消除方法

解决死锁的方法一般可分为预防，避免，检测与恢复。

1. **死锁预防**
2. 预先静态分配法

破坏了“不可剥夺条件”，预先分配资源，保证不等待资源。该方法的问题是降低了对资源的利用率，降低进程的并发程度；有时可能无法预先知道所需的资源

1. 资源有序分配法

破坏了“环路条件”，把资源分类按顺序排列，保证不形成环路。该方法存在的问题是限制进程对资源的请求；由于资源的排序占用系统开销。

1. **死锁避免**

死锁避免算法：银行家算法。

银行家算法对于进程发出的每一个系统可以满足的资源请求命令加以检测，如果发现分配资源后系统进入不安全状态，则不予分配；若分配资源后系统仍处于安全状态，则实施分配

1. **死锁检测**

死锁检测方法对资源的分配不予限制，即允许死锁产生。但系统定时地运行一个死锁检测程序，判断系统是否发生死锁，若检测到有死锁，则设法加以解除。

1. **死锁解除**

死锁解除通常采用如下方法：

1. 资源剥夺法。从一些进程那里强行剥夺足够数量地资源分配给死锁进程
2. 撤销进程法。根据某种策略逐个撤销死锁进程，直至解除死锁为止。

## 3.9 线程的概念

传统的进程有两个属性：可拥有资源的独立单位；可独立调度和分配的基本单位。

引入线程的原因是进程在创建、撤销和切换中，系统必须为之付出较大的时空开销，故在系统中进程的使用不应过高，这就限制了并发程序的提高。所以用线程来，剥离传统的进程。进程作为独立分配资源的单位，线程作为调度和分配的单位。

进程是进程中的一个实体，是被系统独立分配和调度的基本单位。线程基本上可以不拥有资源，只拥有一点运行中必不可少的资源（PC，一组寄存器和栈）。

