1. 进程
2. 定义

进程是程序的以一次执行，该程序可与其他程序并发执行，它是一个动态实体，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。一个进程可以包括多个程序，一个程序也可以被多个进程执行。

1. 组成
2. 概述

进程是在一个上下文的执行环境中运行的，这个环境称为进程的映像，包括CPU中各个通用寄存器的值，进程的内存映像，打开文件的状态以及进程占用资源信息等。进程映像主要是存储映像，当一个进程暂时退出CPU时，它的各种信息都会保存起来称为存储镜像的一部分，当进程被重新调度时，就能够从存储镜像中恢复执行环境。因此，进程映像指的是存储镜像。

进程的存储镜像由以下部分组成：进程控制块（进程状态信息，PCB），进程执行的程序，执行时需要的数据和执行时使用的工作区，这就是进程的组成。工作区指的是执行环境，进程在和心态下运行的工作区为核心栈，在用户态下为用户栈。进程控制块中包含了进程的描述信息和控制信息，是进程存在的标志。

1. 进程控制块

PCB是操作系统用来记录进程详细状态和相关信息的基本数据结构，它和进程是一一对应的，是进程的唯一标志，记录了进程的表示信息，状态信息和控制信息。在内存中以表的形式存在，操作系统对PCB实行集中统一的管理，所有的PCB集中在一个固定存储空间中，构成了PCB表。PCB的数目通常是在操作系统配置完成后确定的，取决于系统最大可并行执行的进程数。

1. 进程基本状态
2. 状态分类

一般有三种状态，运行态，就绪态和阻塞态。运行态表示进程已占有CPU，就绪态表示进程等待系统为其分配CPU，阻塞态表示进程为等待某个事件而挂起。当CPU分配的时间片用完时，或者在中断机制下，有更高优先级的进程进入系统，当前运行的进程进就绪态。

1. 进程队列

将同种状态的PCB排成一个队列，利用指针组成列联表，便于查找，当进程状态变化时，进行出队和入队。

1. 进程控制

主要任务是创建和销毁进程以及实现进程间的状态转换和进程间通信，这就是操作系统内核的功能，内科和是基于硬件的第一层软件扩充，通过执行各种原语操作实现控制功能。原语是指由机器指令构成的可完成特定功能的程序段，它是一个机器指令的集合，在执行时不可中断。

1. 进程相关原语
2. 进程创建原语

有两种方式创建进程，一种是在系统生成时创建的系统进程，另一种经过创建原语产生的，这些进程是非常驻的系统进程和用户进程，原语步骤如下：

系统为进程分配空白的PCB，产生唯一进程标识；

为进程映像分配内存空间；

PCB初始化，PCB参数以及进程初始化就绪状态；

设置相应的链接，插入进程的就绪队列和家族队列中；

1. 进程撤销原语

如果该进程是子进程，只撤销该进程；如果是父进程在，则撤销所有子进程；如果该进程处于运行状态，则中断CPU，保护CPU现场，停止该进程，删除该进程所在的队列标记。具体步骤如下：

在PCB集合中找到要撤销的进程；

归还该进程所占有的资源；

撤销该进程的子进程；

归还PCB空间；

1. 进程阻塞原语

进程的阻塞是自身调用阻塞原语的结果，自愿行为。步骤如下：

中断进程的执行，将CPU状态保存到PCB中；

改写PCB中相应内容，并将进程插入相应的阻塞队列中；

1. 进程唤醒原语

唤醒操作是由其他进程调用唤醒原语完成的，操作系统将阻塞的进程唤醒，进入就绪队列，步骤如下：

从阻塞队列中取出要唤醒进程的PCB；

改写PCB内容，将进程插入就绪队列；

1. 进程调度
2. 概述

进程调度就是按照一定的算法，从就绪队列中选择某个进程占用CPU。

1. 评价调度算法的原则
2. 公平性；
3. 较大的吞吐量；
4. 及时性；
5. 较短的周转时间；
6. 进程调度算法
7. FIFO

先来先执行，直到执行结束或被迫等待才会让出CPU。

1. 基于优先数

每个进程设置一个优先数，可以静态设置优先数，也可以动态设置优先数。进程占用CPU的方式有两种，一种是不可抢占式，如FIFO；另一种是可抢占式，当一个进程运行时被更高优先级的进程抢走了CPU，会使得进程切换频繁，如时间片轮转算法。

1. 时间片轮转

系统规定一个时间长度作为运行进程运行的时间，进程不管执行完成与否，时间片时间一到都需要交出CPU，等待下一次CPU分配。

1. 多级队列算法

引入多个就绪队列，最高优先级队列的进程运行一个时间片，次高队列运行两个时间片，最低优先级运行4个时间片，这样能够保证优先级低的进程能够得到长时间运行。

1. 进程间的相互作用
2. 进程间的关系
3. 直接制约

并发执行的进程之间存在着制约关系，一个进程需要等待另一个进程的执行完成才能够继续执行，这两个进程存在着共同的目的。

1. 间接制约

多个进程竞争只能被一个进程使用的资源，只能通过进程同步机制（互斥）来协调使用顺序。

1. 进程间的同步和互斥
2. 概述

系统中一些需要协同工作的进程之间的关系为同步关系；另一些进程需要争用临界资源而互斥执行，为互斥关系。所谓的临界资源，就是独占资源，设计临界资源的程序段称为临界区或互斥区，各个进程对临界区的访问应该是互斥的。

1. 实现进程互斥的方法
2. 软件法

Dekker算法：

设置整型变量turn，指示允许进入临界区的进程，A进程退出临界区就需要将该变量修改为另一个进程的值。这样，多个进程能够有序进入临界区。

Peterson算法：

除了设置turn，还为每个进程设置标志，指示该进程是否要求进入临界区。

这些方法的缺点是需要不断测试turn的值，阻塞了进程的执行，这种现象为“忙等待”。

1. 硬件法

开关中断指令，也称硬件锁，进程在进入临界区之前先执行“关中断”操作，屏蔽其他中断请求，进程结束后，执行“开中断”，保证有序性。

1. 进程的同步机制
2. 概述

使用信号量以及有关的P，V操作，P表示检测，V表示增量，P操作检测整数变量是否为正值，若不是则阻塞调用进程；V操作唤醒一个阻塞进程恢复执行。信号量表示资源的实体，是一个整型变量，只有P，V操作能够改变。公用信号量用于互斥，初值设为1，私用信号量用于同步，初值设为0或者n，允许拥有它的进程进行P操作。

1. PV操作

P操作相当于对信号量表示资源的申请，所以信号量S-1，如果减1后的结果小于0，说明之前已经没有资源了，资源已被占用，需要等待；如果结果>0,，表示还有资源则进程继续执行，信号量的绝对值表示在等待的进程数。

V操作相当于对信号量表示资源的释放，所以信号量S+1，如果加1后的结果<=0，表示之前还有进程在等待，那么就从等待队列中唤醒一个进程；如果结果>0，表示没有等待的进程，则V操作继续执行。

1. PV操作实现互斥

设定公用信号量为1，那么多个进程就会争用这一个信号量资源，只需要将临界区放置在P(S)和V(S)之间，即可完成两个进程的互斥。

1. PV操作实现同步

两个进程间实现同步操作，每个进程有自己私有的信号量，P操作相当于申请资源，V操作相当于释放资源，同步操作中P,V操作处于不同的进程中，但都是成对出现的。经典案例，生产者和消费者问题。

1. PV操作优缺点

操作原语完备，表达能力强，任何同步和互斥问题都可以解决，但是进程间通信不安全。

1. 管程
2. 概述

管程是解决同步问题的抽象数据类型，指关于共享资源的数据在其上操作的一组过程，弥补了对临界资源管理的不足。管程是一个由过程，变量以及数据结构组成的集合，进程可以调用管程中的过程，但是不能在管程外的过程中直接访问管程中的数据结构。管程的基本思想是集中管理各个进程中的临界区。

管程能够有效地互斥，任何时刻管程中只有一个活跃进程。进入管程时的互斥由编译器负责，使用管程的人不需要知道如何实现的，只需要将所有的临界区转换为管程中的过程即可。

1. 组成

管程名，局部变量，数据结构说明，操作原语，初始化代码。

1. 特点

安全，管程中的共享变量外部看不到，只能由管程中的过程存取；

互斥，任意时刻只有一个进程进入管程；

等待，设置等待队列，对资源进行管理；

1. 管程与进程的区别
2. 设置目的不同

设置管程是为了对共享资源进行管理，进程是程序的一次顺序执行。

1. 数据结构不同

管程定义公用数据结构，进程定义私有数据结构。

1. 存在方式不同

进程有生命周期，是动态变化的。

1. 执行方式不同

管程被进程调用，没有并发性，进程有并发性。