# 1 进程和线程的区别

* 进程：进程是系统进行资源分配和调度的一个独立单位，是系统中的并发执行的单位。
* 线程：线程是进程的一个实体，也是 CPU 调度和分派的基本单位，它是比进程更小的能独立运行的基本单位，有时又被称为轻权进程或轻量级进程。
* 进程是资源分配的最小单位，而线程是 CPU 调度的最小单位。
* 创建进程或撤销进程，系统都要为之分配或回收资源，操作系统开销远大于创建或撤销线程时的开销。
* 不同进程地址空间相互独立，同一进程内的线程共享同一地址空间。一个进程的线程在另一个进程内是不可见的。
* 进程间不会相互影响，而一个线程挂掉将可能导致整个进程挂掉。

# 2 死锁

由于系统中存在一些不可剥夺资源，而当两个或两个以上进程占有自身资源，并请求对方资源时，会导致每个进程都无法向前推进，这就是死锁。

# 3 有了进程为什么还要线程

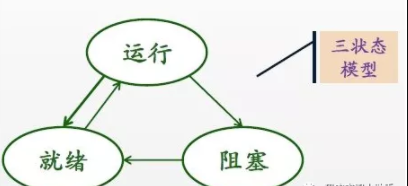
进程可以使多个程序并发执行，以提高资源的利用率和系统的吞吐量，但是其带来了一些缺点：

进程在同一时间只能干一件事情；

进程在执行的过程中如果阻塞，整个进程就会被挂起，即使进程中有些工作不依赖与等待的资源，仍然不会执行。

# 4 进程的状态切换

进程包括三种状态：就绪态、运行态和阻塞态。



* 就绪 —> 执行：对就绪状态的进程，当进程调度程序按一种选定的策略从中选中一个就绪进程，为之分配了处理机后，该进程便由就绪状态变为执行状态；
* 执行 —> 阻塞：正在执行的进程因发生某等待事件而无法执行，则进程由执行状态变为阻塞状态，如进程提出输入/输出请求而变成等待外部设备传输信息的状态，进程申请资源（主存空间或外部设备）得不到满足时变成等待资源状态，进程运行中出现了故障（程序出错或主存储器读写错等）变成等待干预状态等等；
* 阻塞 —> 就绪：处于阻塞状态的进程，在其等待的事件已经发生，如输入/输出完成，资源得到满足或错误处理完毕时，处于等待状态的进程并不马上转入执行状态，而是先转入就绪状态，然后再由系统进程调度程序在适当的时候将该进程转为执行状态；
* 执行 —> 就绪：正在执行的进程，因时间片用完而被暂停执行，或在采用抢先式优先级调度算法的系统中,当有更高优先级的进程要运行而被迫让出处理机时，该进程便由执行状态转变为就绪状态。

# 5 进程间的通信方式有哪些？

进程间通信（IPC，InterProcess Communication）是指在不同进程之间传播或交换信息。IPC 的方式通常有管道（包括无名管道和命名管道）、消息队列、信号量、共享存储、Socket、Streams 等。其中 Socket 和 Streams 支持不同主机上的两个进程 IPC。

**管道**

* 它是半双工的，具有固定的读端和写端；
* 它只能用于父子进程或者兄弟进程之间的进程的通信；
* 它可以看成是一种特殊的文件，对于它的读写也可以使用普通的 read、write 等函数。但是它不是普通的文件，并不属于其他任何文件系统，并且只存在于内存中。

**消息队列**

* 消息队列，是消息的链接表，存放在内核中。一个消息队列由一个标识符 ID 来标识；
* 消息队列是面向记录的，其中的消息具有特定的格式以及特定的优先级；

**信号量**

* 信号量（semaphore）是一个计数器。用于实现进程间的互斥与同步，而不是用于存储进程间通信数据；
* 信号量用于进程间同步，若要在进程间传递数据需要结合共享内存；
* 信号量基于操作系统的 PV 操作，程序对信号量的操作都是原子操作；
* 每次对信号量的 PV 操作不仅限于对信号量值加 1 或减 1，而且可以加减任意正整数；

**共享内存**

* 共享内存（Shared Memory），指两个或多个进程共享一个给定的存储区；
* 共享内存是最快的一种 IPC，因为进程是直接对内存进行存取。

# 6 进程的调度算法

* **先来先服务调度算法**

先来先服务调度算法是一种最简单的调度算法，也称为先进先出或严格排队方案。当每个进程就绪后，它加入就绪队列。当前正运行的进程停止执行，选择在就绪队列中存在时间最长的进程运行。该算法既可以用于作业调度，也可以用于进程调度。先来先去服务比较适合于常作业（进程），而不利于段作业（进程）。

* **时间片轮转调度算法**

时间片轮转调度算法主要适用于分时系统。在这种算法中，系统将所有就绪进程按到达时间的先后次序排成一个队列，进程调度程序总是选择就绪队列中第一个进程执行，即先来先服务的原则，但仅能运行一个时间片。

* **短作业优先调度算法**

短作业优先调度算法是指对短作业优先调度的算法，从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行。 短作业优先调度算法是一个非抢占策略，他的原则是下一次选择预计处理时间最短的进程，因此短进程将会越过长作业，跳至队列头。

* **最短剩余时间优先调度算法**

最短剩余时间是针对最短进程优先增加了抢占机制的版本。在这种情况下，进程调度总是选择预期剩余时间最短的进程。当一个进程加入到就绪队列时，他可能比当前运行的进程具有更短的剩余时间，因此只要新进程就绪，调度程序就能可能抢占当前正在运行的进程。像最短进程优先一样，调度程序正在执行选择函数是必须有关于处理时间的估计，并且存在长进程饥饿的危险。

* **优先级调度算法**

优先级调度算法每次从后备作业队列中选择优先级最髙的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。在进程调度中，优先级调度算法每次从就绪队列中选择优先级最高的进程，将处理机分配给它，使之投入运行。

# 7 死锁的必要条件

* 互斥条件：进程要求对所分配的资源进行排它性控制，即在一段时间内某资源仅为一进程所占用。
* 请求和保持条件：当进程因请求资源而阻塞时，对已获得的资源保持不放。
* 不剥夺条件：进程已获得的资源在未使用完之前，不能剥夺，只能在使用完时由自己释放。
* 环路等待条件：在发生死锁时，必然存在一个进程–资源的环形链。

# 8 怎么预防死锁

互斥条件一般无法破坏

* 破坏请求条件：一次性分配所有资源，这样就不会再有请求了；
* 破坏请保持条件：只要有一个资源得不到分配，也不给这个进程分配其他的资源：
* 破坏不可剥夺条件：当某进程获得了部分资源，但得不到其它资源，则释放已占有的资源；
* 破坏环路等待条件：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反。

# 9 什么是用户态和内核态

* 内核态：处于内核态的 CPU 可以访问任意的数据，包括外围设备，比如网卡、硬盘等，处于内核态的 CPU 可以从一个程序切换到另外一个程序，并且占用 CPU 不会发生抢占情况，一般处于特权级 0 的状态我们称之为内核态。
* 用户态：处于用户态的 CPU 只能受限的访问内存，并且不允许访问外围设备，用户态下的 CPU 不允许独占，也就是说 CPU 能够被其他程序获取。

# 10 一个程序从开始运行到结束的完整过程

**预编译**

主要处理源代码文件中的以“#”开头的预编译指令。处理规则见下

删除所有的#define，展开所有的宏定义。

2、处理所有的条件预编译指令，如“#if”、“#endif”、“#ifdef”、“#elif”和“#else”。

3、处理“#include”预编译指令，将文件内容替换到它的位置，这个过程是递归进行的，文件中包含其他 文件。

删除所有的注释，“//”和“/\*\*/”。

5、保留所有的#pragma 编译器指令，编译器需要用到他们，如：#pragma once 是为了防止有文件被重 复引用。

6、添加行号和文件标识，便于编译时编译器产生调试用的行号信息，和编译时产生编译错误或警告是 能够显示行号。

**编译**

把预编译之后生成的xxx.i或xxx.ii文件，进行一系列词法分析、语法分析、语义分析及优化后，生成相应的汇编代码文件。

词法分析，语法分析，语义分析，优化，目标代码生成。

**汇编**

将汇编代码转变成机器可以执行的指令(机器码文件)

**链接**

将不同的源文件产生的目标文件进行链接，从而形成一个可以执行的程序。链接分为静态链接和动态链接：