1.进程通信方式

1. 管道(匿名管道):

> 1. 速度慢，容量有限，只有父子进程能通讯
> 2. 管道是一种**半双工**的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用。进程的亲缘关系通常是指父子进程关系。

2. FIFO(有名管道):

> 1. 任何进程间都能通讯，但速度慢
> 2. 有名管道也是半双工的通信方式

3. 消息队列:

> 1. 是**有消息的链表**，存放在内核中并由消息队列标识符标识。
> 2. 容量受到系统限制，且要注意第一次读的时候，要考虑上一次没有读完数据的问题

4. 信号量:

> **不能传递复杂消息，只能用来同步**
>
> 1. 信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为**进程间以及同一进程内不同线程之间的同步手段。**

5. 共享内存区： **能够很容易控制容量，速度快，但要保持同步**，比如一个进程在写的时候，另一个进程要注意读写的问题，相当于线程中的线程安全，当然，共享内存区同样可以用作线程间通讯，不过没这个必要，线程间本来就已经共享了同一进程内的一块内存
6. 套接字通信

> 1. 套接口也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同机器间的进程通信。
> 总结:

- 匿名管道（父子进程）
- 有名管道（任意进程）
- 信号量（一般用来同步）
- 套接字（可以用于不同机器之间）
- 共享内存区（容量易控制，速度快，但需要注意同步问题）

- 消息队列（信号量受限，需要注意上一次的读取情况）

2.进程和线程的区别

1. 地址空间： 线程**共享本进程的地址空间**，而进程之间是**独立的地址空间。**
2. 资源：

- 线程**共享本进程的资源如内存**、I/O、cpu等，不利于资源的管理和保护，而进程之间的**资源是独立的**，能很好的进行资源管理和保护。
- 进程是CPU资源分配的基本单位，线程是独立运行和独立调度的基本单位（CPU上真正运行的是线程）。
- 进程拥有自己的资源空间，一个进程包含若干个线程，线程与CPU资源分配无关，多个线程共享同一进程内的资源。

3. 健壮性： **多进程要比多线程健壮**，一个进程崩溃后，在保护模式下不会对其他进程产生影响，但是一个线程崩溃整个进程都死掉。
4. 执行过程：

- 每个独立的进程有一个程序运行的入口、顺序执行序列和程序入口，执行开销大。
- 但是线程不能独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制，执行开销小。
- 线程的调度与切换比进程快很多。
- CPU密集型代码(各种循环处理、计算等等)：使用多进程。IO密集型代码(文件处理、网络爬虫等)：使用多线程

5. 可并发性： 两者均可并发执行。
6. 切换时： 进程切换时，消耗的资源大，效率高。所以涉及到频繁的切换时，使用线程要好于进程。同样如果要求同时进行并且又要共享某些变量的并发操作，只能用线程不能用进程。
7. 其他： 线程是**处理器调度的基本单位**，但是进程不是,进程是资源调度的基本单位。

3.进程、线程、协程的概念

进程：

- 是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**，是一个动态概念，**竞争计算机系统资源的基本单位。** 线程：
- 是进程的一个执行单元，是进程内调度实体。比进程更小的独立运行的基本单位。线程也被称为轻量级进程。 协程：
- 是一种比线程更加轻量级的存在。一个线程也可以拥有多个协程。其执行过程更类似于**子例程**，或者说**不带返回值的函数调用。**

4.协程和线程的区别

1. 协程 协程避免了无意义的调度，由此可以提高性能，但程序员必须自己承担调度的责任。同时，协程也失去了标准线程使用多CPU的能力。
2. 线程

> 相对独立 有自己的上下文 **切换受系统控制**；

3. 协程

> 相对独立 有自己的上下文 **切换由自己控制**，由当前协程切换到其他协程由当前协程来控制。

5.并发与并行

并发：

- 在操作系统中，某一时间段，几个程序在同一个CPU上运行，但在任意一个时间点上，只有一个程序在CPU上运行。
- 当有多个线程时，如果系统只有一个CPU，那么CPU不可能真正同时进行多个线程，CPU的运行时间会被划分成若干个时间段，每个时间段分配给各个线程去执行，一个时间段里某个线程运行时，其他线程处于挂起状态，这就是并发。并发解决了程序排队等待的问题，如果一个程序发生阻塞，其他程序仍然可以正常执行。 并行：
- 当操作系统有多个CPU时，一个CPU处理A线程，另一个CPU处理B线程，**两个线程互相不抢占CPU资源，可以同时进行**，这种方式成为并行。 区别
- 并发只是在宏观上给人感觉有多个程序在同时运行，**但在实际的单CPU系统中，每一时刻只有一个程序在运行，微观上这些程序是分时交替执行。**
- 在多CPU系统中，将这些并发执行的程序分配到不同的CPU上处理，**每个CPU用来处理一个程序，这样多个程序便可以实现同时执行。**

6.进程和线程的区别?

进程

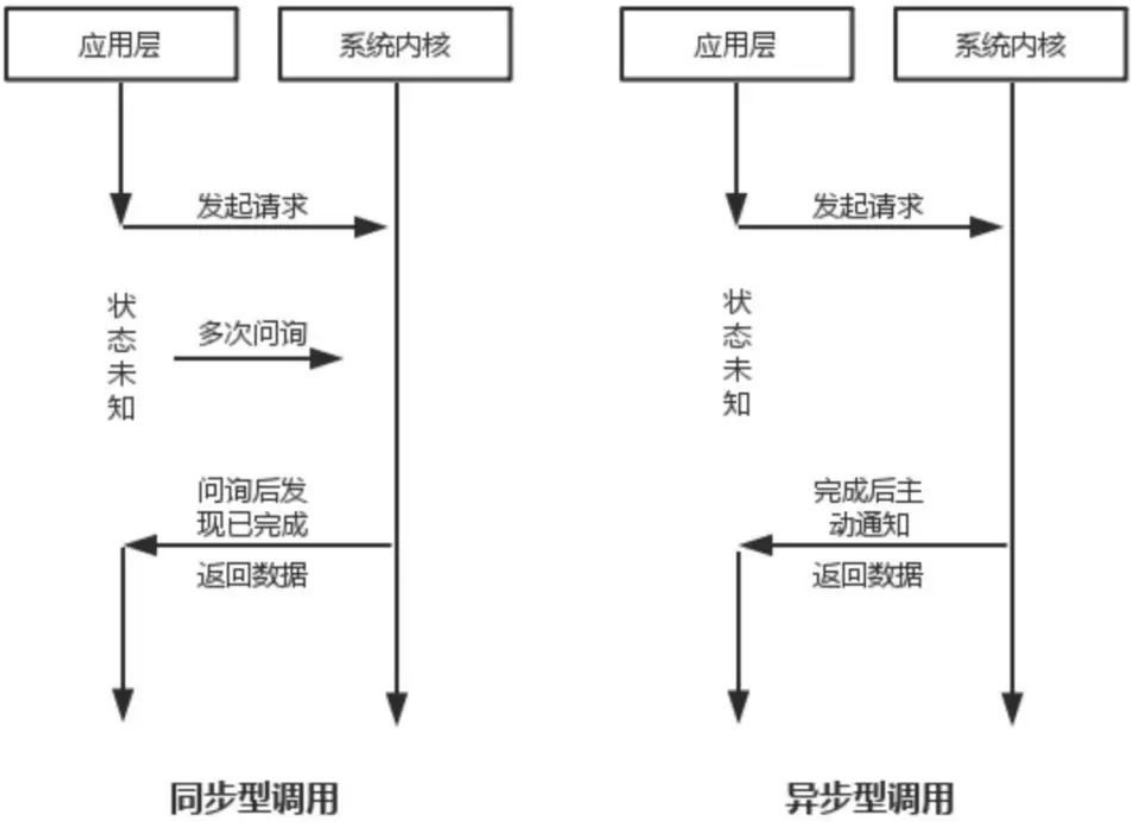
- 一个进程好比是一个程序，它是**资源分配的最小单位。同一时刻执行的进程数不会超过核心数。**不过如果问单核CPU能否运行多进程？答案又是肯定的。单核CPU也可以运行多进程，只不过不是同时的，而是极快地在进程间来回切换实现的多进程。举个简单的例子，就算是十年前的单核CPU的电脑，也可以聊QQ的同时看视频。
- 电脑中有许多进程需要处于「同时」开启的状态，而利用CPU在进程间的快速切换，可以实现「同时」运行多个程序。而进程切换则意味着需要保留进程切换前的状态，以备切换回去的时候能够继续接着工作。所以进程拥有自己的地址空间，全局变量，文件描述符，各种硬件等等资源。操作系统通过调度CPU去执行进程的记录、回复、切换等等。 线程
- 如果说进程和进程之间相当于程序与程序之间的关系，那么线程与线程之间就相当于程序内的任务和任务之间的关系。所以线程是依赖于进程的，也称为「微进程」。它是程序执行过程中的最小单元 。
- 一个程序内包含了多种任务。打个比方，用播放器看视频的时候，视频输出的画面和声音可以认为是两种任务。当你拖动进度条的时候又触发了另外一种任务。拖动进度条会导致画面和声音都发生变化，如果进程里没有线程的话，那么可能发生的情况就是：
    - 拖动进度条->画面更新->声音更新。你会明显感到画面和声音和进度条不同步。
- 但是加上了线程之后，线程能够共享进程的大部分资源，并参与CPU的调度。意味着它能够在进程间进行切换，实现「并发」，从而反馈到使用上就是拖动进度条的同时，画面和声音都同步了。所以我们经常能听到的一个词是「多线程」，就是把一个程序分成多个任务去跑，让任务更快处理。不过线程和线程之间由于某些资源是独占的，会导致锁的问题。

7.阻塞与非阻塞

- 阻塞是指调用线程或者进程被操作系统挂起。
- 非阻塞是指调用线程或者进程不会被操作系统挂起。

8.同步与异步

- 同步是阻塞模式，异步是非阻塞模式。
- 同步就是指一个进程在执行某个请求的时候，若该请求需要一段时间才能返回信息，那么这个进程将会一直等待下去，知道收到返回信息才继续执行下去；
- 异步是指进程不需要一直等下去，而是继续执行下面的操作，不管其他进程的状态。当有消息返回式系统会通知进程进行处理，这样可以提高执行的效率。
- 由调用方盲目主动问询的方式是同步调用，由被调用方主动通知调用方任务已完成的方式是异步调用。看下图



9.协程的定义和好处

协程

- 协程，又称微线程，纤程。英文名Coroutine。一句话说明什么是线程：协程是一种**用户态的轻量级线程。**

- 协程拥有自己的寄存器上下文和栈。协程调度切换时，将寄存器上下文和栈保存到其他地方，在切回来的时候，恢复先前保存的寄存器上下文和栈。因此：
- 协程能保留上一次调用时的状态（即所有局部状态的一个特定组合），每次过程重入时，就相当于进入上一次调用的状态，换种说法：进入上一次离开时所处逻辑流的位置。 协程的好处：
- 无需线程上下文切换的开销
- 无需原子操作锁定及同步的开销 方便切换控制流，简化编程模型 高并发+高扩展性+低成本：一个CPU支持上万的协程都不是问题。所以很适合用于高并发处理。 缺点：
- 无法利用多核资源：协程的本质是个单线程,它不能同时将 单个CPU的多个核用上,协程需要和进程配合才能运行在多CPU上.当然我们日常所编写的绝大部分应用都没有这个必要，除非是cpu密集型应用。
- 进行阻塞（Blocking）操作（如IO时）会阻塞掉整个程序 最佳实践
- 线程和协程推荐在IO密集型的任务(比如网络调用)中使用，而在CPU密集型的任务中，表现较差。
- 对于CPU密集型的任务，则需要多个进程，绕开GIL的限制，利用所有可用的CPU核心，提高效率。
- 所以大并发下的最佳实践就是多进程+协程，既充分利用多核，又充分发挥协程的高效率，可获得极高的性能。