# 协程

协程，又称微线程。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。

优点：

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 "原子操作(atomic operation)是不需要 synchronized"，所谓原子操作是指不会被线程调度机制打断的操作；这种操作一旦开始，就一直运行到结束，中间不会有任何 context switch （切换到另一个线程）。原子操作可以是一个步骤，也可以是多个操作步骤，但是其顺序是不可以被打乱，或者切割掉只执行部分。视作整体是原子性的核心。 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。 进行阻塞（Blocking）操作（如 IO 时）会阻塞掉整个程序。

特性：

必须在只有一个单线程里实现并发 修改共享数据不需加锁 用户程序里自己保存多个控制流的上下文栈 一个协程遇到 IO 操作自动切换到其它协程

在 Python3 中 asyncio 模块没有出来之前，协程可以通过 yield、gevent 来实现，asyncio 模块出来之后，基本都是用 asyncio 来开发

import gevent  
  
def func1():  
 print("我在玩游戏...")  
 gevent.sleep(2)  
 print("切换回去继续玩游戏...")  
  
def func2():  
 print('我要去上厕所。。。')  
 gevent.sleep(1)  
 print('上完厕所了。。。')  
  
gevent.joinall([  
 gevent.spawn(func1),  
 gevent.spawn(func2),  
])

## todo

协程真是太难了，到现在还不会

## 再来亿遍

协程的特点在于是一个线程执行，那和多线程比，协程有何优势？

最大的优势就是协程极高的执行效率。因为子程序切换不是线程切换，而是由程序自身控制，因此，没有线程切换的开销，和多线程比，线程数量越多，协程的性能优势就越明显。

第二大优势就是不需要多线程的锁机制，因为只有一个线程，也不存在同时写变量冲突，在协程中控制共享资源不加锁，只需要判断状态就好了，所以执行效率比多线程高很多。

因为协程是一个线程执行，那怎么利用多核CPU呢？最简单的方法是多进程+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。

## 参考资料

《全面剖析 Python 面试知识点》

https://gitbook.cn/books/5ca40fd11763103ff10b0e43/index.html

https://www.cnblogs.com/russellyoung/p/python-zhi-xie-cheng.html

https://www.cnblogs.com/cheyunhua/p/11017057.html

https://www.jianshu.com/p/7c851145ee4c

https://www.liaoxuefeng.com/wiki/897692888725344/923057403198272