

# 多线程爬虫

## 进程、线程，协程对比

### 定义的不同

**进程**是系统进行资源分配和调度的一个独立单位。一个程序在一个数据集中的一次动态执行过程，可以简单理解为“正在执行的程序”。

**进程一般由程序、数据集、进程控制块三部分组成。**

**程序**用来描述进程要完成哪些功能以及如何完成；**数据集**则是程序在执行过程中所需要使用的资源；**进程控制块**用来记录进程的外部特征，描述进程的执行变化过程，系统可以利用它来控制和管理进程，它是系统感知进程存在的唯一标志。

进程的局限是创建、撤销和切换的开销比较大。

线程是进程的一个实体，也叫轻量级进程，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。**线程自己基本上不拥有系统资源**，只拥有一点在运行中必不可少的资源(如程序计数器，一组寄存器和栈)，但是它**可与同属一个进程的其他的线程共享进程所拥有的全部资源**。

协程，又称微线程，纤程。英文名 Coroutine。

协程是一种用户态的轻量级线程，即协程是由用户程序自己控制调度的。

协程是一个线程执行，但执行有点像多线程

协程执行过程中，在协程内部可中断，然后转而执行别的协程，在适当的时候再返回来接着执行。协程之间不是调用者与被调用者的关系，而是彼此对称、平等的，通过相互协作共同

完成任务。优势：

协程的执行效率极高。

因为协程切换不是线程切换，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。

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。

Python 对用在 generator 中的 yield 可以一定程度上实现协程。通过 yield 方式转移执行权。

### 功能

进程，能够完成多任务，比如 在一台电脑上能够同时运行多个 QQ

线程，能够完成多任务，比如 一个 QQ 中的多个聊天窗口

### 区别

一个程序至少有一个进程,一个进程至少有一个线程.

线程的划分尺度小于进程(资源比进程少)，使得多线程程序的并发性高。

进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率

线程不能够独立执行，必须依存在进程中

### 优缺点

线程和进程在使用上各有优缺点：线程执行开销小，但不利于资源的管理和保护；而进程正相反。

## 队列对象 Queue

python 中的标准库，队列是基本的 FIFO 容器，线程间最常用的交换数据的形式，FIFO 先进先出

**引用：**from queue import Queue

python 原生的 list,dict 等，都是 not thread safe 的。而 Queue，是线程安全的，多线程环境下资源的管理，建议使用队列

## 常用方法

Queue.qsize() 返回队列的大小

Queue.empty() 如果队列为空，返回 True,反之 False

Queue.full() 如果队列满了，返回 True,反之 False

Queue.full 与 maxsize 大小对应

Queue.get([block[, timeout]])获取队列，timeout 等待时间

put(item[, block[, timeout]]) 将 item 放入队列中。

1. 如果可选的参数 block 为 True 且 timeout 为空对象，阻塞调用，无超时（默认的情况）。
2. 如果 timeout 是个正整数，阻塞调用进程最多 timeout 秒，如果一直无空空间可用，抛出 Full 异常（带超时的阻塞调用）。
3. 如果 block 为 False，如果有空闲空间可用将数据放入队列，否则立即抛出 Full 异常

## 示例

#创建一个“队列”对象

from queue import Queue

```
myqueue = Queue(maxsize = 10) #队列中能存放的数据个数的上限
#将一个值放入队列中
myqueue.put(10)
#将一个值从队列中取出
myqueue.get()
```

## 案例：内涵吧多线程爬虫

### 需求

设计多线程爬虫爬取内涵吧

- 1、用三个线程爬取 10 页页面内容，放入队列
- 2、用三个线程解析队列中的页面内容
- 3、把提取的内容存入 json 文件

### 多线程示意图



