Python标准库08 多线程与同步 (threading包)

Python主要通过标准库中的threading包来实现多线程。在当今网络时代，每个服务器都会接收到大量的请求。服务器可以利用多线程的方式来处理这些请求，以提高对网络端口的读写效率。Python是一种网络服务器的后台工作语言 （比如豆瓣网），所以多线程也就很自然被Python语言支持。

（关于多线程的原理和C实现方法，请参考我之前写的Linux多线程与同步，要了解race condition, mutex和condition variable的概念）

多线程售票以及同步

我们使用Python来实现Linux多线程与同步文中的售票程序。我们使用mutex （也就是Python中的Lock类对象） 来实现线程的同步：

```
# A program to simulate selling tickets in multi-thread way
# Written by Vamei

import threading
import time
import os

# This function could be any function to do other chores.
def doChore():
    time.sleep(0.5)

# Function for each thread
```

```
def booth(tid):
    global i
    global lock
    while True:
        lock.acquire()                  # Lock; or wait if
other thread is holding the lock
        if i != 0:
            i = i - 1                   # Sell tickets
            print(tid,':now left:',i)  # Tickets left
            doChore()                   # Other critical
operations
        else:
            print("Thread_id",tid," No more tickets")
            os._exit(0)                 # Exit the whole
process immediately
        lock.release()                  # Unblock
        doChore()                       # Non-critical
operations

# Start of the main function
i     = 100                             # Available ticket
number
lock = threading.Lock()                 # Lock (i.e., mutex)

# Start 10 threads
for k in range(10):
    new_thread = threading.Thread(target=booth,args=(k,))
# Set up thread; target: the callable (function) to be run,
args: the argument for the callable
    new_thread.start()
# run the thread
```

我们使用了两个全局变量，一个是i，用以储存剩余票数；一个是lock对象，用于同步线程对i的修改。此外，在最后的for循环中，我们总共设置了10个线程。每个线程都执行booth()函数。线程在调用start()方法的时候正式启动 （实际上，计算机中最多会有11个线程，因为主程序本身也会占用一个线程）。Python使用threading.Thread对象来代表线程，用threading.Lock对象来代表一个互斥锁 (mutex)。

有两点需要注意：

- 我们在函数中使用global来声明变量为全局变量，从而让多线程共享i和lock（在C语言中，我们通过将变量放在所有函数外面来让它成为全局变量）。如果不这么声明，由于i和lock是不可变数据对象，它们将被当作一个局部变量(参看Python动态类型)。如果是可变数据对象的话，则不需要global声明。我们甚至可以将可变数据对象作为参数来传递给线程函数。这些线程将共享这些可变数据对象。
- 我们在booth中使用了两个doChore()函数。可以在未来改进程序，以便让线程除了进行i=i-1之外，做更多的操作，比如打印剩余票数，找钱，或者喝口水之类的。第一个doChore()依然在Lock内部，所以可以安全地使用共享资源(critical operations，比如打印剩余票数)。第二个doChore()时，Lock已经被释放，所以不能再去使用共享资源。这时候可以做一些不使用共享资源的操作 (non-critical operation，比如找钱、喝水)。我故意让doChore()等待了0.5秒，以代表这些额外的操作可能花费的时间。你可以定义的函数来代替doChore()。

OOP创建线程

上面的Python程序非常类似于一个面向过程的C程序。我们下面介绍如何通过面向对象 (OOP，object-oriented programming，参看Python面向对象的基本概念和Python面向对象的进一步拓展) 的方法实现多线程，其核心是继承threading.Thread类。我们上面的for循环中已经利用了threading.Thread()的方法来创建一个Thread对象，并将函数booth()以及其参数传递给改对象，并调用start()方法来运行线程。OOP的话，通过修改Thread类的run()方法来定义线程所要执行的命令。

```
# A program to simulate selling tickets in multi-thread way
# Written by Vamei

import threading
import time
import os
```

```
# This function could be any function to do other chores.
def doChore():
    time.sleep(0.5)

# Function for each thread
class BoothThread(threading.Thread):
    def __init__(self, tid, monitor):
        self.tid           = tid
        self.monitor = monitor
        threading.Thread.__init__(self)
    def run(self):
        while True:
            monitor['lock'].acquire()
# Lock; or wait if other thread is holding the lock
            if monitor['tick'] != 0:
                monitor['tick'] = monitor['tick'] - 1
# Sell tickets
                print(self.tid,':now
left:',monitor['tick'])    # Tickets left
                doChore()
# Other critical operations
            else:
                print("Thread_id",self.tid," No more
tickets")
                os._exit(0)
   # Exit the whole process immediately
            monitor['lock'].release()
   # Unblock
            doChore()
   # Non-critical operations

# Start of the main function
monitor = {'tick':100, 'lock':threading.Lock()}

# Start 10 threads
for k in range(10):
    new_thread = BoothThread(k, monitor)
    new_thread.start()
```

我们自己定义了一个类BoothThread，这个类继承自thread.Threading类。然后我们把上面的booth()所进行的操作统统放入到BoothThread类的run()方法中。注意，我们没有使用全局变量声明global，而是使用了一个词典monitor存放全局变量，然后把词典作为参数传递给线程函数。由于词典是可变数据对象，所以当它被传递给函数的时候，函数所使用的依然是同一个对象，相当于被多个线程所共享。这也是多线程乃至于多进程编程的一个技巧（应尽量避免上面的global声明的用法，因为它并不适用于windows平台）。

上面OOP编程方法与面向过程的编程方法相比，并没有带来太大实质性的差别。

其他

threading.Thread对象： 我们已经介绍了该对象的start()和run()，此外：

- join()方法，调用该方法的线程将等待直到改Thread对象完成，再恢复运行。这与进程间调用wait()函数相类似。

下面的对象用于处理多线程同步。对象一旦被建立，可以被多个线程共享，并根据情况阻塞某些进程。请与Linux多线程与同步中的同步工具参照阅读。

threading.Lock对象： mutex，有acquire()和release()方法。

threading.Condition对象： condition variable，建立该对象时，会包含一个Lock对象（因为condition variable总是和mutex一起使用）。可以对Condition对象调用acquire()和release()方法，以控制潜在的Lock对象。此外：

- wait()方法，相当于cond_wait()
- notify_all()，相当与cond_broadcast()
- nofify()，与notify_all()功能类似，但只唤醒一个等待的线程，而不是全部

threading.Semaphore对象： semaphore，也就是计数锁(semaphore传统意义上是一种进程间同步工具，见Linux进程间通信)。创建对象的时候，可以传递一个整数作为计数上限（sema = threading.Semaphore(5)）。它与Lock类似，也有Lock的两个方法。

threading.Event对象： 与threading.Condition相类似，相当于没有潜在的

Lock保护的condition variable。对象有True和False两个状态。可以多个线程使用wait()等待，直到某个线程调用该对象的set()方法，将对象设置为True。线程可以调用对象的clear()方法来重置对象为False状态。

练习

参照Linux多线程与同步中的condition variable的例子，使用Python实现。同时考虑使用面向过程和面向对象的编程方法。

更多的threading的内容请参考：

http://docs.python.org/library/threading.html

总结

threading.Thread

Lock, Condition, Semaphore, Event

标签: Python

Vamei
关注 - 26
粉丝 - 4985
荣誉：推荐博客
+加关注

« 上一篇：Linux的概念与体系
» 下一篇：Python标准库09 当前进程信息 (os包)

posted @ 2012-10-11 18:27 Vamei 阅读(24689) 评论(20) 编辑 收藏

评论列表

#1楼 2012-10-11 23:13 JeffWong

挺好 最近也在看Python多线程

#2楼 2012-10-23 00:50 ZZB

按照你的例子，用condition写了个练习，不知道是不是这个意思？！

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

import time, os
import threading

class BoothThread(threading.Thread):
    def __init__(self, tid, cond):
        self.tid = tid
        self.cond = cond
        threading.Thread.__init__(self)

    def run(self):
        self.cond.acquire()

        global num
        num += 1
        if num <= 10:
            self.cond.wait()
            print self.tid, 'drink beer'
        elif num == 11:
            self.cond.notify_all()

        self.cond.release()


if __name__ == '__main__':
    num = 0
    cond = threading.Condition()
    for tid in xrange(100):
        bt = BoothThread(tid, cond)
        bt.start()
```

支持(0) 反对(0)

#3楼[楼主] 2012-10-23 09:36 Vamei

@ ZZB

需要获得和释放Condition中的Lock，不然还是会有race condition的可能性。

支持(0) 反对(0)

#4楼 2012-10-23 11:00 ZZB

@ Vamei

博主，我这里不是有self.cond.acquire()和self.cond.release()吗？这两个就是获得锁和释放锁，难道这样写的逻辑还不够？请指教～

支持(0) 反对(0)

#5楼[楼主] 2012-10-23 11:12 Vamei

@ ZZB

shit,我刚才看得不仔细。错了错了，抱歉啊。

支持(0) 反对(0)

#6楼 2012-11-17 16:04 ma6174

很好的文章！操作系统里面学了点关于进程控制和通信的文章，一直不知道怎么实现，看了楼主的文章受益匪浅！谢谢！

支持(0) 反对(0)

#7楼 2013-02-26 11:16 冬火虫子

我尝试用面向过程的方法做了一下练习，参考了ZZB写的代码

```
1  import os, time, threading
2
3  def doChore():
4      time.sleep(2)
5
6  def drink(tid, cond):
7      global num
8      cond.acquire()
9
10     num +=1
11     if num <=10:
12         cond.wait()
13         print tid, "drink beer"
14     elif num == 11:
15         cond.notify_all()
16
17     cond.release()
18     doChore()
19
20
21 if __name__ == '__main__':
22     num = 0
23     cond = threading.Condition()
24     for tid in range(100):
```

```
25          new_thread = threading.Thread(target = drink,args = (tid,cond))
26          new_thread.start()
```

支持(0) 反对(0)

#8楼 2013-02-26 11:20 冬火虫子

请问：threading.Thread.__init__(self)在子类中的这句代码是做什么的呢？没有这句会有什么情况发生呢？

支持(0) 反对(0)

#9楼[楼主] 2013-02-26 13:53 Vamei

@ 冬火虫子

由于继承了Thread类，所以初始化的时候执行该父类的初始化函数。

文档中要求调用该函数，我没有试过不使用这一句会发生什么。

支持(0) 反对(0)

#10楼 2013-07-21 00:05 海浪轻风

面向过程的写法

```
1  import threading
2  import time
3  import os
4  def doCore():
5          time.sleep(0.5)
6  def booth(tid):
7          global i
8          global cond
9          cont.acquire()
10         i+=1
11         if(i<=10):
12                 cont.wait()
13                 print('drink beer')
14                 doCore()
15         elif(i==11):
16                 cont.notify_all()
17         cont.release()
18
19 i=0
20 cont=threading.Condition()
21 for k in range(100):
22         new_thread=threading.Thread(target=booth,args=(k,))
23         new_thread.start()
```

面向对象版本

```
import threading,os,time
def doChore():
        time.sleep(0.5)
class WorkThread(threading.Thread):
        def __init__(self,id,monitor):
                self.id=id
                self.monitor=monitor
                threading.Thread.__init__(self)
        def run(self):
                monitor['cont'].acquire()
                monitor['num']=monitor['num']+1
                if(monitor['num']<=10):
                        monitor['cont'].wait()
                        print('drink beer')
                        doChore()
                elif(monitor['num']==11):
                        monitor['cont'].notify_all()
                monitor['cont'].release()
                doChore()



monitor={'num':0,'cont':threading.Condition()}
for i in range(100):
        workThread=WorkThread(i,monitor)
        workThread.start()
```

支持(0) 反对(0)

#11楼 2013-11-05 11:15 wanyao

@vamei

为什么面向对象的版本运行起来线程是有顺序的？不科学阿，求解释～

(0, 'now left:', 99)

(1, 'now left:', 98)

(2, 'now left:', 97)

(3, 'now left:', 96)

(4, 'now left:', 95)

(5, 'now left:', 94)

(6, 'now left:', 93)

(7, 'now left:', 92)
(8, 'now left:', 91)
(9, 'now left:', 90)
(0, 'now left:', 89)
(1, 'now left:', 88)
(2, 'now left:', 87)
(3, 'now left:', 86)
(4, 'now left:', 85)
(5, 'now left:', 84)
(6, 'now left:', 83)
(7, 'now left:', 82)
(8, 'now left:', 81)
(9, 'now left:', 80)
(0, 'now left:', 79)
(1, 'now left:', 78)
(2, 'now left:', 77)
(3, 'now left:', 76)
(4, 'now left:', 75)

支持(1) 反对(0)

#12楼 2014-01-16 12:30 icy_123

PS C:\py> python thread.py
(0, ':now left:', 99)
Exception in thread Thread-1:
Traceback (most recent call last):
File "C:\Program Files (x86)\Python276\lib\threading.py", line 810, in __bootstrap_inner
self.run()
File "thread.py", line 23, in run
monitor['tick'].release()
AttributeError: 'int' object has no attribute 'release'
请问这个是因为windows的系统里的Int没有这个属性么

支持(0) 反对(1)

#13楼 2014-03-24 11:07 圣徒甜茶

@ icy_123
monitor['tick'] 是个整数对象，怎么release。。。
lock（）才能release，代码里对应的monitor['lock']

支持(0) 反对(0)

#14楼 2014-05-05 17:33 xbmiracle

@ Vamei
类似于java中的重载，感觉这是固定的格式，可以使用
super(BoothThread,self).__init__()代替Threading.Thread.__init__()

支持(0) 反对(0)

#15楼 2014-05-11 22:33 Yu Zi

楼主
为什么用monitor['lock'].acquire()，而不用self.monitor['lock'].acquire()，按照你的例子，没有用到self.monitor，虽然self.monitor赋值了。

支持(1) 反对(0)

#16楼 2014-06-01 15:36 滚出碗里

Linux中对于 信号默认的处理时 终止进程，比如ctrl +c
上面的买票(oop)例子 也是没有对于信号进行过处理的

那为什么 当我运行那个例子的时候 按ctrl +c程序没有终止 而是继续在跑？？？？？？？

支持(0) 反对(0)

#17楼 2014-06-01 15:47 滚出碗里

字典里的 那个lock 你定义成 'lock:threadng.Lock()'

这样当你每次 调用这个lock 会不会重新 生成一个 新的锁对象？？

比如：
lock.acquire()
lock.release()

上面 上锁和开锁 是对同一个对象经行操作？

支持(0) 反对(0)

#18楼 2014-06-01 15:54 滚出碗里

实验了下 发现 这个所其实是在定义字典的时候就已经生成了，后面就直接是使用这个对象。

支持(0) 反对(0)

#19楼 2014-08-19 14:39 花瓣奶牛

好文章，不错不错，原来可以使用字典的技术来代替全局变量的使用啊。。太棒了

支持(0) 反对(0)

#20楼 2015-12-03 12:39 irunner

@ Vamei

同问，这个方法调用是只有继承threading.Thread才有还是说其他的子类继承父类都需要在init里显式调用父类的init。

如果不调用会报错：

raise RuntimeError("thread.__init__() not called")

RuntimeError: thread.__init__() not called

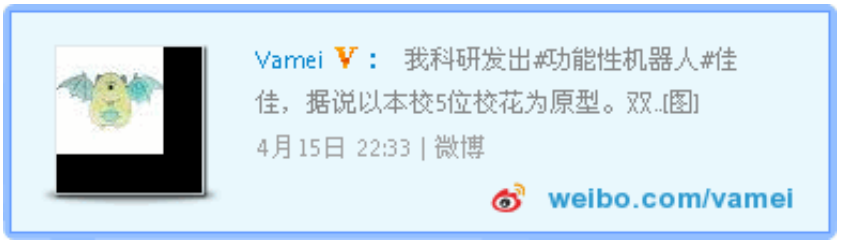
支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

公告

你好，这里是Vamei，一名编程爱好者。我在博客里写了**Python/Linux/**网络协议**/**算法**/Java/**数据科学系列文章，从这里开始阅读。非常期待和你的交流。



我的微博

下列教程已经做成电子出版物，内容经过修订，也方便离线阅读：

协议森林

欢迎阅读我写的其他书籍：

现代小城的考古学家

天气与历史的相爱相杀

随手拍光影

昵称：Vamei
园龄：4年1个月
荣誉：推荐博客
粉丝：4985
关注：26
+加关注

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

Python(61)

Java(42)

大数据(22)

Linux(17)

网络(16)

算法(15)
文青(14)
技普(9)
系列索引(6)
开发工具(4)
更多

系列文章

Java快速教程
Linux的概念与体系
Python快速教程
数据科学
协议森林
纸上谈兵: 算法与数据结构

积分与排名

积分 - 659668
排名 - 122

最新评论

1. Re:Java基础11 对象引用
受教!

--MisslLost

2. Re:Python快速教程
看评论区一片喝彩！看来我得在此扎营了！

--测试小蚂蚁

3. Re:Python进阶06 循环对象
好好地列表解析变成了表推导

--ashic

4. Re:“不给力啊，老湿！”：RSA加密与破解
感谢楼主精彩分享

--worldball

5. Re:概率论04 随机变量
你写的这一系列太棒了，刚加入博客园就在你这里学到了，我要转载过去学习一下

--yixius

6. Re:Python基础03 序列

挺好的教程、、、、

--王小拽的号

7. Re:Python进阶07 函数对象

def func(x,y): print x**ydef test(f,a,b): print 'test' print f(a,b)test (func,3,2)

输出的内容:tes.....

--M-edea

8. Re:Python进阶02 文本文件的输入输出

@coderXT换行符：\n...

--行者之印

9. Re:数据科学

博主啊，这里是一枚即将大二的计算机新人，大一学了python，java，还有一些算法，数据结构，图论了，感觉我对数学又一些反感，但是听说离散数学对计算机专业的很重要，不知道怎么去学比较好呢，我想像您写.....

--Acokil

10. Re:为什么要写技术博

楼主是用自己自定义的模板吗？在博客园里找不到这种风格的blog模板？

--行者之印

11. Re:来玩Play框架01 简介

挖煤哥,我补充了一下Windows下的搭建play框架,希望有点帮助,谢谢!

--Sungeek

12. Re:来玩Play框架07 静态文件

@helper.form(action = routes.Application.upload, 'enctype ->

"multipart/form-data") {--action = rout.....

--quxiaozha

13. Re:来玩Play框架07 静态文件

该记录将/assets/下的URL，对应到项目的/public文件夹内的文件。比如在项目的/public/images/test.jpg，就可以通过/assests/images/test.jpg这一.....

--quxiaozha

14. Re:来玩Play框架06 用户验证

支持挖煤哥~~~

--quxiaozha

15. Re:“不给力啊，老湿！”：RSA加密与破解

@maanshancss请你仔细阅读了这个文章再来评价。...

--Vamei

推荐排行榜

1. “不给力啊，老湿！”：RSA加密与破解(218)
2. Python快速教程(140)
3. 野蛮生长又五年(91)
4. Java快速教程(88)
5. 协议森林01 邮差与邮局 (网络协议概观)(79)
6. 为什么要写技术博(71)
7. 编程异闻录(54)
8. 博客一年：心理之旅(49)
9. 协议森林08 不放弃 (TCP协议与流通信)(45)
10. Python快速教程 尾声(43)
11. 协议森林(42)
12. Java基础01 从HelloWorld到面向对象(42)
13. Python基础08 面向对象的基本概念(40)
14. 一天能学会的计算机技术(34)
15. 博客第二年，杂谈(33)