Python标准库10 多进程初步（multiprocessing包）

我们已经见过了使用subprocess包来创建子进程，但这个包有两个很大的局限性：1） 我们总是让subprocess运行外部的程序，而不是运行一个Python脚本内部编写的函数。2） 进程间只通过管道进行文本交流。以上限制了我们将subprocess包应用到更广泛的多进程任务。(这样的比较实际是不公平的，因为subprocessing本身就是设计成为一个shell，而不是一个多进程管理包)

threading和multiprocessing

(请尽量先阅读Python多线程与同步)

multiprocessing包是Python中的多进程管理包。与threading.Thread类似，它可以利用multiprocessing.Process对象来创建一个进程。该进程可以运行在Python程序内部编写的函数。该Process对象与Thread对象的用法相同，也有start(), run(), join()的方法。此外multiprocessing包中也有Lock/Event/Semaphore/Condition类 (这些对象可以像多线程那样，通过参数传递给各个进程)，用以同步进程，其用法与threading包中的同名类一致。所以，multiprocessing的很大一部份与threading使用同一套API，只不过换到了多进程的情境。

但在使用这些共享API的时候，我们要注意以下几点：

- 在UNIX平台上，当某个进程终结之后，该进程需要被其父进程调用wait，否则进程成为僵尸进程(Zombie)。所以，有必要对每个Process对象调用join()方法 (实际上等同于wait)。对于多线程来说，由于只有一个进程，所以不存在

此必要性。

- multiprocessing提供了threading包中没有的IPC(比如Pipe和Queue)，效率上更高。应优先考虑Pipe和Queue，避免使用Lock/Event/Semaphore/Condition等同步方式 (因为它们占据的不是用户进程的资源)。
- 多进程应该避免共享资源。在多线程中，我们可以比较容易地共享资源，比如使用全局变量或者传递参数。在多进程情况下，由于每个进程有自己独立的内存空间，以上方法并不合适。此时我们可以通过共享内存和Manager的方法来共享资源。但这样做提高了程序的复杂度，并因为同步的需要而降低了程序的效率。

Process.PID中保存有PID，如果进程还没有start()，则PID为None。

我们可以从下面的程序中看到Thread对象和Process对象在使用上的相似性与结果上的不同。各个线程和进程都做一件事：打印PID。但问题是，所有的任务在打印的时候都会向同一个标准输出(stdout)输出。这样输出的字符会混合在一起，无法阅读。使用Lock同步，在一个任务输出完成之后，再允许另一个任务输出，可以避免多个任务同时向终端输出。

```
# Similarity and difference of multi thread vs. multi
process
# Written by Vamei

import os
import threading
import multiprocessing

# worker function
def worker(sign, lock):
    lock.acquire()
    print(sign, os.getpid())
    lock.release()

# Main
print('Main:',os.getpid())

# Multi-thread
```

```
record = []
lock  = threading.Lock()
for i in range(5):
    thread = threading.Thread(target=worker,args=
('thread',lock))
    thread.start()
    record.append(thread)

for thread in record:
    thread.join()

# Multi-process
record = []
lock = multiprocessing.Lock()
for i in range(5):
    process = multiprocessing.Process(target=worker,args=
('process',lock))
    process.start()
    record.append(process)

for process in record:
    process.join()
```

所有Thread的PID都与主程序相同，而每个Process都有一个不同的PID。

(练习：使用mutiprocessing包将Python多线程与同步中的多线程程序更改为多进程程序)

Pipe和Queue

正如我们在Linux多线程中介绍的管道PIPE和消息队列message queue，multiprocessing包中有Pipe类和Queue类来分别支持这两种IPC机制。Pipe和Queue可以用来传送常见的对象。

1) Pipe可以是单向(half-duplex)，也可以是双向(duplex)。我们通过mutiprocessing.Pipe(*duplex=False*)创建单向管道 (默认为双向)。一个进

程从PIPE一端输入对象，然后被PIPE另一端的进程接收，单向管道只允许管道一端的进程输入，而双向管道则允许从两端输入。

下面的程序展示了Pipe的使用：

```
# Multiprocessing with Pipe
# Written by Vamei

import multiprocessing as mul

def proc1(pipe):
    pipe.send('hello')
    print('proc1 rec:',pipe.recv())

def proc2(pipe):
    print('proc2 rec:',pipe.recv())
    pipe.send('hello, too')

# Build a pipe
pipe = mul.Pipe()

# Pass an end of the pipe to process 1
p1   = mul.Process(target=proc1, args=(pipe[0],))
# Pass the other end of the pipe to process 2
p2   = mul.Process(target=proc2, args=(pipe[1],))
p1.start()
p2.start()
p1.join()
p2.join()
```

这里的Pipe是双向的。

Pipe对象建立的时候，返回一个含有两个元素的表，每个元素代表Pipe的一端(Connection对象)。我们对Pipe的某一端调用send()方法来传送对象，在另一端使用recv()来接收。

2) Queue与Pipe相类似，都是先进先出的结构。但Queue允许多个进程放入，多个进程从队列取出对象。Queue使用mutiprocessing.Queue(*maxsize*)创建，maxsize表示队列中可以存放对象的最大数量。

下面的程序展示了Queue的使用：

```
# Written by Vamei
import os
import multiprocessing
import time
#==================
# input worker
def inputQ(queue):
    info = str(os.getpid()) + '(put):' + str(time.time())
    queue.put(info)

# output worker
def outputQ(queue,lock):
    info = queue.get()
    lock.acquire()
    print (str(os.getpid()) + '(get):' + info)
    lock.release()
#====================
# Main
record1 = []    # store input processes
record2 = []    # store output processes
lock  = multiprocessing.Lock()    # To prevent messy print
queue = multiprocessing.Queue(3)

# input processes
for i in range(10):
    process = multiprocessing.Process(target=inputQ,args=
(queue,))
    process.start()
    record1.append(process)

# output processes
for i in range(10):
    process = multiprocessing.Process(target=outputQ,args=
```

```
(queue,lock))
    process.start()
    record2.append(process)

for p in record1:
    p.join()

queue.close()  # No more object will come, close the queue

for p in record2:
    p.join()
```

一些进程使用put()在Queue中放入字符串，这个字符串中包含PID和时间。另一些进程从Queue中取出，并打印自己的PID以及get()的字符串。

总结

Process, Lock, Event, Semaphore, Condition

Pipe, Queue

标签: Python

Vamei
关注 - 26
粉丝 - 4985

« 上一篇：Python标准库09 当前进程信息 (os包)
» 下一篇：Python标准库11 多进程探索 (multiprocessing包)

posted @ 2012-10-12 22:25 Vamei 阅读(49196) 评论(36) 编辑 收藏

评论列表

#1楼 2012-10-14 01:41 micfan

匆匆看过，有待实践

支持(0) 反对(0)

#2楼[楼主] 2012-10-14 19:28 Vamei

@ micfan

多进程和多线程的话可以放在网络应用里实践。

支持(0) 反对(0)

#3楼 2012-11-14 23:22 msheng.yeb

你好，我没有明白为什么这里要用锁

```
# output worker
def outputQ(queue,lock):
info = queue.get()
lock.acquire()
print (str(os.getpid()) + '(get):' + info)
lock.release()
```

不是说每个进程都有自己独立的内存空间？

支持(0) 反对(0)

#4楼[楼主] 2012-11-15 10:11 Vamei

@ msheng.yeb

因为这些进程共享一个stdout。多个进程同时向stdout输出的话，输出结果会混合在一起。用锁，可以让一个进程输出之后，再由另一个进程输出。

支持(0) 反对(0)

#5楼 2012-11-15 23:13 msheng.yeb

@ Vamei

明白了，stdout是共享的

支持(0) 反对(0)

#6楼[楼主] 2012-11-16 09:40 Vamei

@ msheng.yeb

应该说是继承的。你可以看一下我的Linux教程里的fork机制。

支持(0) 反对(0)

#7楼 2012-11-17 23:47 msheng.yeb

@ Vamei

不叫继承吧。fork的话，就是把父进程的内容拷贝一份给子进程，包括stdout，然后父进程和子进程有各自的stdout。我猜想，是由于这个stdout指向了同一个终端，所以不锁的话，有可能造成输出混乱

支持(0) 反对(0)

#8楼[楼主] 2012-11-18 10:06 Vamei

@ msheng.yeb

嗯，原理就是这样。

支持(0) 反对(0)

#9楼 2013-01-14 10:30 moose

为什么我跑上面的例子，跑一次电脑死一次.....

支持(0) 反对(0)

#10楼[楼主] 2013-01-14 11:00 Vamei

@ moose

少几个进程试一试？

支持(0) 反对(0)

#11楼 2013-01-15 23:09 bells

练习：我是这样写的：

```
import time
import os
import multiprocessing

def doChore():
    time.sleep(0.5)

def booth(pid):
    global i
    global lock
    while True:
        lock.acquire()
        if i != 0:
            i = i - 1
            print os.getpid(), "  ", pid, ': now left:', i
            doChore()
        else:
            print "Thread_id", pid, "no more tickers"
```

```
            os._exit(0)
        lock.release()
        doChore()

i = 100
lock = multiprocessing.Lock()

for k in range(10):
    process = multiprocessing.Process(target=booth, args=(k, ))
    process.start()
```

运行输出：

```
2819 0 : now left: 99
2820 1 : now left: 99
2821 2 : now left: 99
2822 3 : now left: 99
2823 4 : now left: 99
2824 5 : now left: 99
2825 6 : now left: 99
2826 7 : now left: 99
2827 8 : now left: 99
2828 9 : now left: 99
2819 0 : now left: 98
2820 1 : now left: 98
2821 2 : now left: 98
2822 3 : now left: 98
2823 4 : now left: 98
2824 5 : now left: 98
2825 6 : now left: 98
2826 7 : now left: 98
2827 8 : now left: 98
2828 9 : now left: 98
2819 0 : now left: 97
2820 1 : now left: 97
2821 2 : now left: 97
```

...

觉得输出有问题。。比如99不应该输出10次？

楼主上面的代码有问题吗？

支持(0) 反对(0)

#12楼 2013-02-26 14:35 冬火虫子

运行了一下第一个示例程序，但是每次运行总是会死机，而且通过任务管理器观察发现，进程越来越多，这是什么情况呢？

支持(0) 反对(0)

#13楼[楼主] 2013-02-26 15:12 Vamei

@ 冬火虫子

你分别运行下多进程和多线程，看是哪一部分出问题了。我是用Linux测试的，没有windows的电脑。

支持(0) 反对(0)

#14楼[楼主] 2013-02-26 15:20 Vamei

@ bells

我觉得这是多进程下Lock共享的问题。

你尝试将Lock作为参数传递给子进程试一试。

支持(0) 反对(0)

#15楼 2013-04-18 10:15 skyla

为什么写队列的时候不加锁？难道不会产生竞态？

支持(0) 反对(0)

#16楼[楼主] 2013-04-18 13:31 Vamei

@ skyla

你是说put会发生竞态么？

支持(0) 反对(0)

#17楼 2013-04-18 14:13 skyla

对，是不是队列本身就支持原子操作的？ 楼主我做了下练习，代码如下：

```
1 import multiprocessing
2 import time
3 import os
4
5 def doChore():
```

```
    time.sleep(0.5)

def booth(tid, block, bqueue):
    global i
    global lock
    while True:
        block.acquire()
        i = int(bqueue.get())
        if i != 0:
            i = i - 1
            print(tid,':now left:',i)
            bqueue.put(str(i))
            doChore()
        else:
            print("Thread_id",tid," No more tickets")
            os._exit(0)
        block.release()
        doChore()

record = []
lock = multiprocessing.Lock()
queue = multiprocessing.Queue(3)
queue.put(str(30))

for k in range(10):
    new_process = multiprocessing.Process(target=booth,args=(k,lock,queue))
    new_process.start()
    record.append(new_process)

for p in record:
    p.join()

queue.close()
```

运行输出：

```
root@skyla-virtual-machine:/home/skyla/temp# python3 test.py
5 :now left: 29
3 :now left: 28
2 :now left: 27
1 :now left: 26
0 :now left: 25
```

4 :now left: 24
6 :now left: 23
8 :now left: 22
7 :now left: 21
9 :now left: 20
5 :now left: 19
9 :now left: 18
2 :now left: 17
1 :now left: 16
0 :now left: 15
4 :now left: 14
6 :now left: 13
4 :now left: 12
7 :now left: 11
3 :now left: 10
7 :now left: 9
9 :now left: 8
2 :now left: 7
9 :now left: 6
0 :now left: 5
9 :now left: 4
0 :now left: 3
9 :now left: 2
5 :now left: 1
9 :now left: 0
Thread_id 7 No more tickets
不知道为什么程序完成后就卡住了，不能自动退出，尝试了用sys.exit()也不行，还请楼主帮忙分析下

支持(0) 反对(0)

#18楼[楼主] 2013-04-18 14:33 Vamei

@ skyla
队列是存储多个对象的。当调用put的时候，Python就将对象放入队列。
这里，总票数是一个单一对象，队列并不适用。

支持(0) 反对(0)

#19楼 2013-06-03 11:49 Jack.R.S

楼主，不知道你用过multiprocessing的Pool没有，我用map_async这个方法的时候，有个问题，该方法第一个参数为函数，后面的是一个迭代对象，如p.map_async(init_sock, [(server_sock, SIZE), (server_sock, SIZE)])，我这有调用init_sock的时候，发现init_sock没法传递sock对象，如果我吧server_sock换成int对象，就可以了，单独调用init_sock是没问题的。很奇怪！init_sock原型def init_sock((argv1, argv2))

支持(0) 反对(0)

#20楼[楼主] 2013-06-05 16:41 Vamei

@ Jack.R.S

我没这么用过。有空了研究一下。

支持(0) 反对(0)

#21楼 2013-09-10 11:05 天楚

vamei，你好

看了你的博客有一段时间的了。

抱着不求甚解的态度看到了这一段（我觉得我学不来编程，也许这只是自己给自己的借口吧。没什么东西是不需要积累的，但是我觉得看不到什么进步）

我学习python主要是用作主机的日常维护。

但是看到多进程这一步的时候，我到我们服务器上做测试。结果发现python 2.4竟然没有multiprocess这个模块。网上找了下python升级的资料，都挺复杂的。还不能把原先的python删除之类的。

也许我太懒，只图"一招鲜吃遍天"了

支持(0) 反对(0)

#22楼 2013-12-13 17:23 bg2bkk

http://my.oschina.net/u/593413/blog/88573

或许楼上死机的哥们可以看看这个帖子。

不过linux下没有死机，一开始都没有

支持(0) 反对(0)

#23楼 2014-02-13 18:43 M2014

我把这个函数中的锁注释掉了，又添加了更多的打印语句，同时把 sign 参数也补充了id以示区别：

```
def worker(sign, lock):
#lock.acquire()
print(sign, os.getpid())
print(sign, os.getpid())
print(sign, os.getpid())
print(sign, os.getpid())
print(sign, os.getpid())
#lock.release()
```

程序运行后的打印结果并没有乱……试了好多次，难道标准输出有同步处理？

全部代码如下：

```
1  import os
2  import threading
3  import multiprocessing
4
5  def woker(sign, lock):
6          print(sign, os.getpid())
7          print(sign, os.getpid())
8          print(sign, os.getpid())
9          print(sign, os.getpid())
10         print(sign, os.getpid())
11
12 print('Main', os.getpid())
13
14 #multithread
15 record = []
16 lock = threading.Lock()
17 for i in range(20):
18         sig = 'thread_%d' % i
19         thread = threading.Thread(target=woker, args=(sig, lock))
20         thread.start()
21         record.append(thread)
22
23 for thread in record:
24         thread.join()
```


#24楼 2014-03-28 18:01 Alex-Zeng

@ Vamei
’用mutiprocessing.Queue(maxsize)创建，maxsize表示队列中可以存放对象的最大数量’

最大数量作何解释呢，10个放入multiprocessing.Queue(3)中为什么没有报错呢？

支持(0) 反对(0)

#25楼 2014-04-13 17:36 安静从容
学习了，今天学习了Threading和multiprocessing
这两者一个是多线程，一个是多进程，那哪个平时会优先考虑使用呢？

支持(0) 反对(0)

#26楼[楼主] 2014-04-14 10:06 Vamei
@ 安静从容
这个比较难讲。多线程主要特征是可以共享内存。可是有的系统不支持多进程或多线程。所以是一个综合的决定。

支持(0) 反对(0)

#27楼 2014-06-02 22:05 滚出碗里
关于 第一个例子 最好是在进程函数里加上 sleep() 这样对于各个经常并行处理 效果看起来明显一点

支持(0) 反对(0)

#28楼 2014-06-09 14:36 平和的心
Pipe类和Queue类用的都是基于管道的，即os.pipe()。Queue跟消息队列通信不完全一样，它只是允许多读多写而已，只有亲缘关系的进程才能使用，不相关的进程用不了。

支持(0) 反对(0)

#29楼 2014-07-21 16:22 齐柏林飞艇
"multiprocessing应优先考虑Pipe和Queue，....避免使用Lock/Event/Semaphore/Condition等同步方式(因为它们占据的不是用户进程的资源)" 这里有点疑问. mutiprocessing.Lock已经实现了进程共享,而博主说占据的不是用户进程资源. 另外一个问题,mutiprocessing.Lock和Queue比性能上哪个好一点?

支持(0) 反对(0)

#30楼 2014-11-21 11:31 amelie_920

vamei,你好，初学者想问一下最后一个程序为什么只有输出设置Lock，输入不设置呢？

支持(0) 反对(0)

#31楼 2015-12-27 22:02 菜鸟GG

python 3.4 创建进程，

proc = multiprocessing.Process(target=worker)

创建失败，是怎么回事，难道不应该这么用吗

支持(0) 反对(0)

#32楼 2015-12-31 09:21 ching126

```
1  # Filename: myqueue.py
2
3  import os
4  import multiprocessing
5  import time
6
7  def inputQ(queue):
8      info = str(os.getpid()) + '(put):' + str(time.time())
9      queue.put(info)
10
11 def outputQ(queue,lock):
12     info = queue.get()
13     lock.acquire()
14     print(str(os.getpid()) + '(get):' + info)
15     lock.release()
16
17 record1 = []
18 record2 = []
19 lock = multiprocessing.Lock()
20 queue = multiprocessing.Queue(3)
21
22 for i in range(10):
23     process = multiprocessing.Process(target=inputQ,args=(queue,))
24     process.start()
25     record1.append(process)
26
27 for i in range(10):
28     process = multiprocessing.Process(target=outputQ,args=(queue,lock))
29     process.start()
30     record2.append(process)
31
32 for p in record1:
```

```
    p.join()

for p in record2:
    p.join()
```

这段代码我在win7 64bit下运行没有输出东东，标准库10这一节的多进程代码都没有输出，多线程的有输出正常的，不知道这是为什么呢，求教啊，我用的是python2.7，先在这里谢谢大家了，小菜鸟求教，

支持(0) 反对(0)

#33楼 2015-12-31 09:22 ching126

没报错也没有输出，求教啊

支持(0) 反对(0)

#34楼 2016-01-11 15:35 xiaosanyu

@ ching126

引用

没报错也没有输出，求教啊

```
if not queue.empty():
    info = queue.get()
```

加这句看看

支持(0) 反对(0)

#35楼 2016-01-11 15:38 xiaosanyu

```
if not queue.empty():
info = queue.get()
```

支持(0) 反对(0)

#36楼 2016-03-16 15:25 一小步

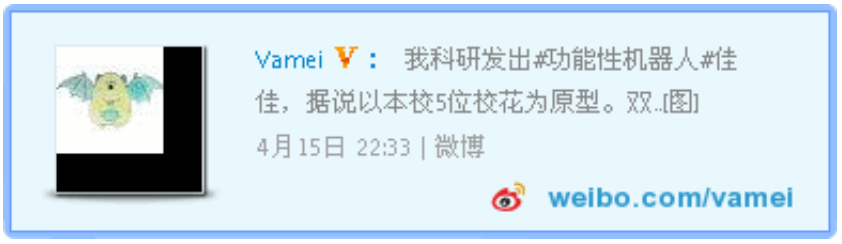
正如我们在Linux多线程中介绍的管道PIPE和消息队列message queue

这里的多线程不是应该是多进程吗，我把Linux多线程看了一遍没看到PIPE啊

支持(0) 反对(0)

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

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)