

一 前言

温习python 多进程语法的时候，对 join的理解不是很透彻，本文通过代码实践来加深对 join()的认识。

multiprocessing 是python提供的跨平台版本的多进程模块。multiprocessing可以充分利用多核，提升程序运行效率。multiprocessing支持子进程,通信和共享数据,执行不同形式的同步,提供了Process、Queue、Pipe、Lock等组件。不过今天重点了解 join。后续文章会逐步学习介绍其他组件或者功能。

二 动手实践

join()方法可以在当前位置阻塞主进程，带执行join()的进程结束后再继续执行主进程的代码逻辑。

代码语言： javascript

复制

```
# encoding: utf-8
"""
author: yangyi@youzan.com
time: 2019/7/30 11:20 AM
func:
"""

from multiprocessing import Process
import os
import time

def now():
    return str(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()))

def func_1(name):
    print(now() + ' Run child process %s (%s)...' % (name, os.getpid()))
    time.sleep(4)
    print(now() + ' Stop child process %s (%s)...\n' % (name, os.getpid()))

def func_2(name):
    print(now() + ' Run child process %s (%s)...' % (name, os.getpid()))
    time.sleep(8)
    print(now() + ' hello world!')
    print(now() + ' Stop child process %s (%s)...\n' % (name, os.getpid()))

if __name__ == '__main__':
    print ('Parent process %s.' % os.getpid())
    p1 = Process(target=func_1, args=('func_1',))
    p2 = Process(target=func_2, args=('func_2',))
    print now() + ' Process start.'
    p1.start()
    p2.start()
    p1.join()
    p2.join()
    print now() + ' Process end .'
```

输出结果

```

→ test git:(master) x python join_test.py
Parent process 75993.
2019-07-30 23:25:37 Process start.
2019-07-30 23:25:37 Run child process func_1 ,pid is 76034...
2019-07-30 23:25:37 Run child process func_2 , pid is 76035...
2019-07-30 23:25:39 Stop child process func_1 ,pid is 76034...
2019-07-30 23:25:41 hello world!
2019-07-30 23:25:41 Stop child process func_2 , pid is 76035...
2019-07-30 23:25:41 Process end .
→ test git:(master) x

```

结果显示

主进程的 Process end .是在func1 和func2 结束之后才打印出来的。

2.2 去掉 join() 函数

代码语言: javascript

复制

```

if __name__ == '__main__':
    print ('Parent process %s.' % os.getpid())
    p1 = Process(target=func_1, args=('func_1',))
    p2 = Process(target=func_2, args=('func_2',))
    print now() + ' Process start.'
    p1.start()
    p2.start()
    print now() + ' Process end .'

```

结果如下:

```

Parent process 76106.
2019-07-30 23:27:50 Process start.
2019-07-30 23:27:50 Process end .
2019-07-30 23:27:50 Run child process func_1 ,pid is 76147...
2019-07-30 23:27:50 Run child process func_2 , pid is 76148...
2019-07-30 23:27:52 Stop child process func_1 ,pid is 76147...
2019-07-30 23:27:54 hello world!
2019-07-30 23:27:54 Stop child process func_2 , pid is 76148...
→ test git:(master) x

```

结果显示主线程 "Process end" 紧跟着 "Process start",然后是 func_1 func_2 的动作。而不是等func_1 func_2执行完才执行。

2.3 去掉func_2 的 join()

代码语言: javascript

复制


```

if __name__ == '__main__':
    print ('Parent process %s.' % os.getpid())
    p1 = Process(target=func_1, args=('func_1',))
    p2 = Process(target=func_2, args=('func_2',))
    print now() + ' Process start.'
    p1.start()
    p2.start()
    p1.join() ### 在p1 执行完之后 。不等待p2 执行, 主进程结束。
    print now() + ' Process end .'

```

结果如下:

```
→ test git:(master) x python join_test.py
Parent process 76244.
2019-07-30 23:32:59 Process start.
2019-07-30 23:32:59 Run child process func_1 ,pid is 76285...
2019-07-30 23:32:59 Run child process func_2 , pid is 76286...
2019-07-30 23:33:01 Stop child process func_1 ,pid is 76285...
2019-07-30 23:33:01 Process end .
2019-07-30 23:33:03 hello world!
2019-07-30 23:33:03 Stop child process func_2 , pid is 76286...
→ test git:(master) x
```

 yangyidba

结果显示主线程 "Process end"在func_1 执行结束之后输出而没有等待func_2 执行完毕。

2.4 小结

利用多线程时，一般都先让子线程调用start()，然后再去调用join()，让主进程等待子进程结束才继续走后续的逻辑。

思考题

能不能每个子进程调用start() 之后，然后直接调用join() 类似:

```
p1.start()p1.join()p2.start()p2.join()
```