

原 Python多进程 - 实现多进程的几种方式

2018年01月05日 15:31:27 Money多多 阅读数：10614

版权声明：http://blog.csdn.net/topleeyap https://blog.csdn.net/topleeyap/article/details/78981848

## 方式一: os.fork()

```
1 # -*- coding:utf-8 -*-
2
3 """
4 pid=os.fork()
5     1. 只用在Unix系统中有效，Windows系统中无效
6     2.fork函数调用一次，返回两次：在父进程中返回值为子进程id，在子进程中返回值为0
7 """
8
9 import os
10
11 pid=os.fork()
12
13 if pid==0:
14     print("执行子进程，子进程pid={pid},父进程ppid={ppid}".format(pid=os.getpid(),ppid=os.getppid()))
15 else:
16     print("执行父进程，子进程pid={pid},父进程ppid={ppid}".format(pid=pid,ppid=os.getpid()))
```

## 方式二: 使用multiprocessing模块: 创建Process的实例，传入任务执行函数作为参数

```
1 # -*- coding:utf-8 -*-
2
3 """
4 Process常用属性与方法:
5     name:进程名
6     pid: 进程id
7     run(), 自定义子类时覆写
8     start(), 开启进程
9     join(timeout=None), 阻塞进程
10    terminate(),终止进程
11    is_alive(), 判断进程是否存活
12 """
13
14 import os,time
15 from multiprocessing import Process
16
17 def worker():
18     print("子进程执行中>>> pid={0},ppid={1}".format(os.getpid(),os.getppid()))
19     time.sleep(2)
20     print("子进程终止>>> pid={0}".format(os.getpid()))
21
22 def main():
23     print("主进程执行中>>> pid={0}".format(os.getpid()))
24
25     ps=[]
26     # 创建子进程实例
27     for i in range(2):
28         p=Process(target=worker,name="worker"+str(i),args=())
29         ps.append(p)
30
31     # 开启进程
32     for i in range(2):
33         ps[i].start()
34
35     # 阻塞进程
36     for i in range(2):
37         ps[i].join()
38
39     print("主进程终止")
40
41 if __name__ == '__main__':
```

## 方式二: 使用multiprocessing模块: 派生Process的子类, 重写run方法

```

1  # -*- coding:utf-8 -*-
2
3  import os,time
4  from multiprocessing import Process
5
6  class MyProcess(Process):
7      def __init__(self):
8          Process.__init__(self)
9
10     def run(self):
11         print("子进程开始>>> pid={0},ppid={1}".format(os.getpid(),os.getppid()))
12         time.sleep(2)
13         print("子进程终止>>> pid={}".format(os.getpid()))
14
15 def main():
16     print("主进程开始>>> pid={}".format(os.getpid()))
17     myp=MyProcess()
18     myp.start()
19     # myp.join()
20     print("主进程终止")
21
22 if __name__ == '__main__':
23     main()

```



2



## 方式四: 使用进程池Pool

```

1  # -*- coding:utf-8 -*-
2
3  import os,time
4  from multiprocessing import Pool
5
6  def worker(arg):
7      print("子进程开始执行>>> pid={},ppid={},编号{}".format(os.getpid(),os.getppid(),arg))
8      time.sleep(0.5)
9      print("子进程终止>>> pid={},ppid={},编号{}".format(os.getpid(),os.getppid(),arg))
10
11 def main():
12     print("主进程开始执行>>> pid={}".format(os.getpid()))
13     ps=Pool(5)
14     for i in range(10):
15         # ps.apply(worker,args=(i,))          # 同步执行
16         ps.apply_async(worker,args=(i,))      # 异步执行
17
18     # 关闭进程池, 停止接受其它进程
19     ps.close()
20     # 阻塞进程
21     ps.join()
22     print("主进程终止")
23
24 if __name__ == '__main__':
25     main()

```



想对作者说点什么

## Python中使用多进程来实现并行处理的方法小结

阅读数 2777

进程和线程是计算机软件领域里很重要的概念, 进程和线程有区别, 也有着密切的联系, 先来辨析一下这两个概念: ... [博文](#) 来自: [云中寻雾的博客](#)