

## 树莓派如何开启多线程

### 多任务编程

除了计算性能和图形显示以外，树莓派区别于 Arduino 的一大特点就是运行多任务操作系统。通过多任务系统用户可以同时执行多个互相独立的程序（任务），来完成不同的操作。

利用 Python 的多任务编程可以方便地实现并行运算，同时充分利用树莓派的多个内核。当然这里面有一些是真的并行操作，还有通过分时轮流使用 CPU 来实现的“伪并行”。

### 多线程编程

多线程操作的特点是简单易用，可用于处理 CPU 占用率不高的任务。虽然一个进程中可以建立多个线程，但由于同一个 Python 进程只能目前利用一个 CPU 内核，因此只能利用树莓派 25% 的 CPU 资源。这里例举两种多线程模块和 threading。thread 模块是比较底层的模块，threading 模块对 thread 做了一些包装，以方便调用。需要注意的是 Python3.x 中已经不再支持 thread 模块，请使用 threading 实现多线程操作。

thread 模块编程示例：

```
import thread
import time
```

```
def thread1():
    while True:
        print("\nThis is thread1!")
        time.sleep(1)

def thread2():
    while True:
        print("\nThis is thread2!")
```

```
time.sleep(2)
```

```
thread.start_new_thread(thread1, ())
```

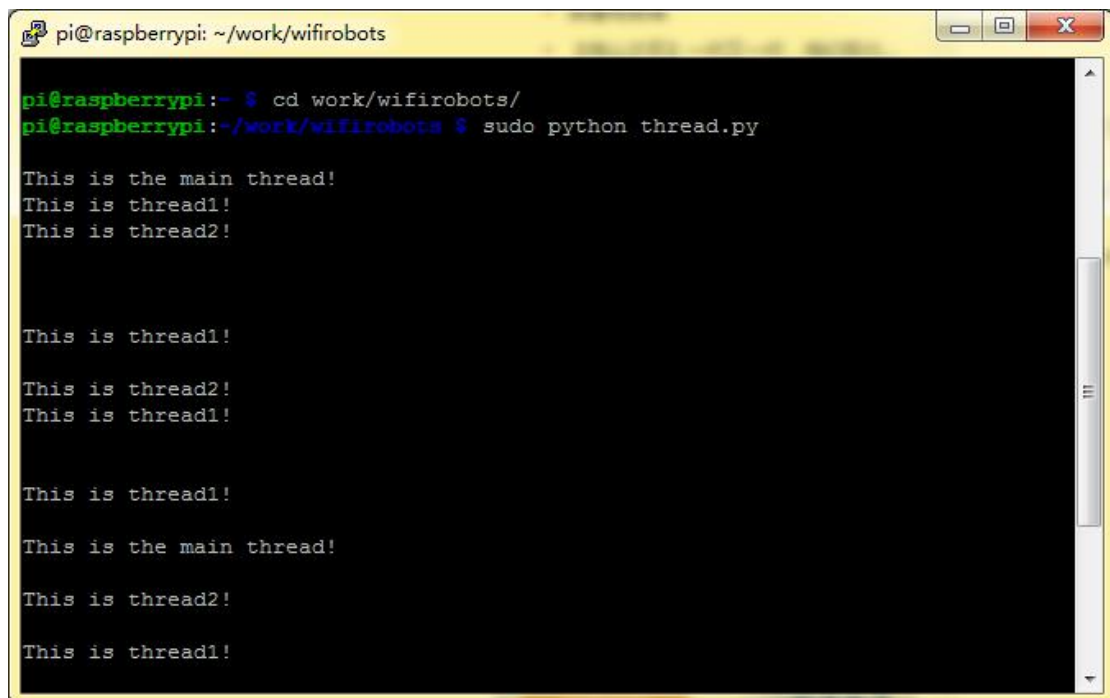
```
thread.start_new_thread(thread2, ())
```

```
while True:
```

```
    print("\nThis is the main thread!")
```

```
    time.sleep(4)
```

运行效果：



```
pi@raspberrypi: ~/work/wifirobots
pi@raspberrypi:~$ cd work/wifirobots/
pi@raspberrypi:~/work/wifirobots$ sudo python thread.py

This is the main thread!
This is thread1!
This is thread2!


This is thread1!

This is thread2!
This is thread1!


This is thread1!

This is the main thread!

This is thread2!

This is thread1!
```

什么时候需要用到多线程呢？比如说：我们需要用树莓派获取一个传感器的数据，但是这个传感器获取数据有一定延时，我们不想让主线程中的程序一直等待这个数据获取到，我们希望主线程可以在等待的这段时间里面做其他事情，等数据获取到的时候我们在去处理这个数据。

```
import thread
```

```
import time
```

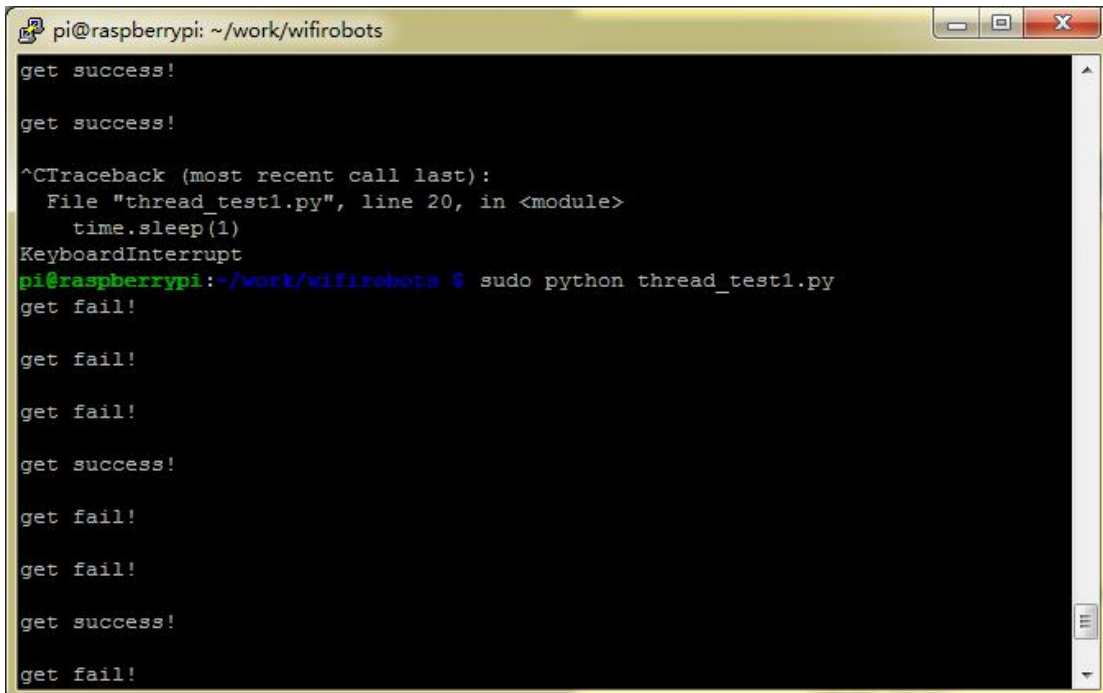
```
global flag
```

```
flag = 0
```

```
def thread1():  
    global flag  
    while True:  
        time.sleep(3)  
        flag = 1  
  
thread.start_new_thread(thread1, ())
```

```
while True:  
    if flag==1:  
        print("get success!\n")  
        flag = 0  
    else:  
        print("get fail!\n")  
        time.sleep(1)
```

运行结果：



```
pi@raspberrypi: ~/work/wifirobots  
get success!  
get success!  
^CTraceback (most recent call last):  
  File "thread_test1.py", line 20, in <module>  
    time.sleep(1)  
KeyboardInterrupt  
pi@raspberrypi:~/work/wifirobots $ sudo python thread_test1.py  
get fail!  
get fail!  
get fail!  
get success!  
get fail!  
get fail!  
get success!  
get fail!
```