### python之多线程编程二

前面我们介绍了thread模块实现的多线程，但thread 模块级别低，并且不支持守护线程这个概念，当主线程退出时所有的子线程也被终止了。所以我们可以使用更高级别，功能更全面的threading模块来管理线程。

threading 模块中有个Thead类，这个模块支持守护线程，主线程将在所有非守护现场退出后才退出，所以只要设置好thread.setDaemon=True，来检测线程的状态就可以达到目的。

下面是使用Treading模块中的Thread类实现的多线程，结果如下：



代码如下：

#!/usr/bin/env python

# coding:utf-8

import threading

from time import ctime,sleep

def f1():

for i in range(4):

print u'看书...',ctime()

sleep(1)

#lock.release()

def f2():

for i in range(2):

print u'听音乐...',ctime()

sleep(1)

threads = []

t1=threading.Thread(target=f1)

threads.append(t1)

t2=threading.Thread(target=f2)

threads.append(t2)

if \_\_name\_\_=='\_\_main\_\_':

print u'开始时间',ctime()

for t in threads:

t.setDaemon(True)

t.start()

for t in threads:

t.join()

print u'结束',ctime()

在上面的例子中，当所有的线程都分配完成之后，通过调用每个线程的start()方法开始执行，相比于管理一组锁（分配、获取、释放、检查锁状态）而言，只需要每个线程调用join()的方法，用join()方法等待线程的结束。

Python学习交流群：548377875