§3.4 Python并发程序设计

徐悦甦(Yueshen Xu)

ysxu@xidian.edu.cn

软件工程系

西安电子科技大学

本节提纲

西安电子科技大学 XIDIAN UNIVERSITY

- Python语言基础
- Python基础语法
- Python面向对象编程

Python语言

- **Python并发程序设计**（Concurrency）
  - Python多线程机制
  - Python中线程的创建

Python并发编程

关键字： Python | 基础语法 | 变量与运算 | 并发编程

Python多线程编程

西安电子科技大学 XIDIAN UNIVERSITY

- 参考方法一：将要执行的函数作为参数传递给threading.Thread()

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import threading, time
def func(n):
    global count
    time.sleep(0.1)
    for i in range(n):
        count += 1
#def ends
if __name__==' __main__' :
    count = 0
    threads = []
```

```
for i in range(5):
    threads.append(threading.Thread(target
    =func, args=(1000,)))

for t in threads:
    t.start()

time.sleep(5)
print( 'count:' ,count)
#main ends
```

Python多线程编程

西安电子科技大学
XIDIAN UNIVERSITY

- 代码分析
  - threading.Thread(target=func, args=(1000,))
    - func 为函数名，args 为函数参数（必须以元组的形式传递）；
  - t.start(): 启动函数，等待操作系统调度；
  - 函数运行结束，线程也就结束；
  - time.sleep(): 线程进入睡眠，处于IO 阻塞状态
- 结果分析
  - 以上例子创建了5 个线程去执行 func 函数
  - 获得的结果可能是 5000，但也有时候会出现错误

Python多线程编程

- 参考方法二：继承 threading.Thread()类，并重写run()（推荐使用）

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import threading, time

class myThread(threading.Thread):
    def __init__(self, n):
        threading.Thread.__init__(self)
        self.myThread_n = n

    def run(self):
        global count
        for i in range(self.myThread_n):
            count += 1
#def ends
```

```
if __name__==’ __main__’ :
    count = 0
    threads = []
    for i in range(5):
        threads.append(myThread(1000))
    for t in threads:
        t.start()
    time.sleep(5)
    print( ‘count:’ ,count)
```

Python多线程编程

西安电子科技大学
XIDIAN UNIVERSITY

线程同步锁互斥控制

```
class myThread(threading.Thread):
    def __init__(self, n):
        threading.Thread.__init__(self)
        self.myThread_n = n
    def run(self):
        global count
        for i in range(self.myThread_n):
            count += 1
    #def ends
```

```
class myThread(threading.Thread):
    def __init__(self, n):
        threading.Thread.__init__(self)
        self.myThread_n = n
    def run(self):
        global count
        for i in range(self.myThread_n):
            __Temp = count
            time.sleep(0.0001)
            count = __Temp + 1
    #def ends
```

Python多线程编程

西安电子科技大学
XIDIAN UNIVERSITY

线程同步锁互斥控制

```
import threading, time
class myThread(threading.Thread):
    def __init__(self, n):
        threading.Thread.__init__(self)
        self.myThread_n = n
    def run(self):
        global count
        for i in range(self.myThread_n):
            if lock.acquire():
                __Temp = count
                time.sleep(0.0001)
                count = __Temp + 1
                lock.release()
```

```
if __name__=' __main__' :
    count = 0
    #同步锁，也称互斥量
    lock = threading.Lock()
    threads = []
    for i in range(5):
        threads.append(myThread(1000))
    for t in threads:
        t.start()
    time.sleep(5)
    print( 'count:' ,count)
```

Python多线程编程

西安电子科技大学
XIDIAN UNIVERSITY

线程同步锁互斥控制

- 代码分析
  - lock = threading.Lock()：创建锁；
  - b. lock.acquire([timeount])：请求锁定，
    - 如果设定了timeout，则在超时后通过返回值可以判断是否得到了锁，
    - 从而可以进行一些其他的处理；
  - c. lock.release()：释放锁定

Python多线程编程

西安电子科技大学 XIDIAN UNIVERSITY

- threading：信号量
  - 信号量用来控制线程并发数的
    - BoundedSemaphore 或Semaphore 管理一个内置的计数器
      - 每当调用acquire()时-1，
      - 调用release() 时+1
    - 计数器不能小于0，当计数器为0 时，acquire()将阻塞线程至同步锁定状态，直到其他线程调用release()
    - BoundedSemaphore 与Semaphore 的唯一区别在于
      - 前者将在调用release()时检查计数器的值是否超过了计数器的初始值
      - 如果超过将抛出一个异常

Python多线程编程

西安电子科技大学 XIDIAN UNIVERSITY

- threading：信号量

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import threading, time
class myThread(threading.Thread):
        def run(self):
            if semaphore.acquire():
              #注意观察semaphore
              print(self.name)
              time.sleep(5)
              semaphore.release()
```

```
if __name__==’ __main__’ :
     semaphore =
    threading.Semaphore(5)
    threads = []
    for i in range(100):
        threads.append(myThread())
    for t in threads:
        t.start()
```

Python多线程编程

西安电子科技大学 XIDIAN UNIVERSITY

- threading：其他函数
  - threading.Timer()
    - Thread 的派生类，用于在指定时间后调用一个方法
  - threading.join()
    - 使得一个线程可以等待另一个线程执行结束后再继续运行
  - threading.isAlive()
    - 判断线程是否还在运行中
  - threading.isDaemon()
    - 返回线程的daemon 标志
  - threading.getName()
    - 返回线程名
  - threading.current_thread()
    - 返回当前线程句柄
  - threading.enumerate()
    - 返回正在运行的线程列表
  - threading.activeCount()
    - 返回正在运行的线程数量

第四次上机

西安电子科技大学 XIDIAN UNIVERSITY

- 时间与地点
  - 6月27号， G楼346、348机房
- 主题：Python并发编程
- 题目
  - 与Java并发编程题目相同
    - 题目一：四个售票窗口同时出售30张电影票
    - 参考思路
      - 票数要使用同一个静态值
      - 为保证不会出现卖出同一个票数，要Python多线程同步锁
      - 创建一个售票窗口类BoxOffice，继承 threading.Thread ，重写run方法，在run方法里面执行售票操作
      - 售票要使用同步锁：即有一个站台卖这张票时，其他站台要等这张票卖完

第四次上机

西安电子科技大学
XIDIAN UNIVERSITY

- 题目二
  - 两个人张三与李四，通过一个同一个账户，张三在柜台取钱，李四在ATM机取钱
- 参考思路
  - 创建一个Bank类
  - 创建一个张三类，代表在柜台取钱
  - 创建一个李四类，代表在ATM
  - 创建主方法的调用类

第四次上机

西安电子科技大学 XIDIAN UNIVERSITY

- 要求
  - 完成两道题目的程序编写，
  - 独立完成实验报告，发送电子版，可以是word，也可以pdf
  - 实验报告提交地址，同作业
    - xdsepc2018@163.com
    - 邮件命名规则：“实验报告四+学号+姓名”
    - 附件命名规则：“实验报告四+学号+姓名”
    - 上交时间：至7.6号星期五，十天时间
    - 实验报告格式：见《并行计算实验报告结构及要求》

附上机整体安排

西安电子科技大学
XIDIAN UNIVERSITY

- 上机时间（四次，16学时）
  - 按周数：第11、12、14、17周的星期三
  - 按日期：5月16号，5月23号，6月6号，6月27号
  - 全部为星期三晚上，18：30至21：30，每次三个小时
  - 地点：G楼346、348机房
  - 内容
    - 第一次（5.16）：Java并发
    - 第二次（5.23）：Java并行
    - 第三次（6.6）：Python基础编程
    - 第四次（6.27）：Python并发编程

附上机整体安排

西安电子科技大学 XIDIAN UNIVERSITY

- 星期三晚上6点半开始，至9点半结束

学期		G346, 348	上学期20180316																					
月		2月	3月				4月				5月					6月				7月				8
周		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
日		26	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30
课期		|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
星期	次	4	11	18	25	1/4	8	15	22	29	6/5	13	20	27	3/6	10	17	24	1/7	8	15	22	29	5/8
一	上										五							端						
	下										一							午						
	晚										节							节						
二	上										五	中												
	下										一	考												
	晚										节	试												
三	上									运		中												
	下									动		考												
	晚									会		试	并	并		并			并					
四	上						清			运														
	下						明			动														
	晚						节			会														

西安电子科技大学
XIDIAN UNIVERSITY

§4.1 并行算法设计

http://web.xidian.edu.cn/ysxu/

结构 编程 **算法** 应用