西安电子科技大学 XIDIAN UNIVERSITY | 软件学院 School of Software

并 行 计 算

课程实验报告

实验名称：Python 并发编程

任课教师：徐悦甦

课程班级：15 级 云计算方向

学号姓名：15130130273 石明皓

提交日期：2018 年 6 月 27 日

软件学院本科生《并行计算》

课程实验报告

一、实验名称

第 4 次实验：Python 并发编程

二、实验日期

2018 年 6 月 27 日 软件学院实验室 G346

三、实验学生

15130130273 石明皓

四、实验目的

本次实验通过展示 2 个简单的多线程应用场景，编写了实现其业务逻辑的 Python 多线程程序。学习了通过继承 threading.Thread 类，然后重载 run 方法来实现多线程，以及学会了 Python 多线程锁的简单应用，理解了基本 Python 多线程编程思想，掌握了 Python 基础并发程序设计的方法。

五、实验内容

题目一：

四个售票窗口同时出售 30 张电影票。

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。

六、程序思路、结构

题目一：

电影票的票数使用同一个静态值，不同售票窗口对象操作的均为该静态变量；

为了不出现不同柜台卖出同一张票的情况，要用到 Python 多线程同步锁，需要使用 threading.Lock()创建一个 lock 对象；

创建售票窗口类 BoxOffice 继承 threading.Thread 类，该类构造时使用 threading.Thread.__init__(self)实现线程的初始化，同时使用 self.setName()赋予售票窗口号，在 BoxOffice 类中重写 run()方法，在 run()方法中进行售票操作；

进行售票操作时使用同步锁，即：任意一个窗口正在出售某张票时，其他窗口必须先等待该窗口卖出这张票，完成其完整售票流程。

每个窗口对象的 run()方法循环执行，对某张电影票操作前 lock.acquire()加锁，操作完成后 lock.release()解锁；

创建 4 个线程模拟 4 个售票窗口，使用 start()方法启动线程。

题目二：

创建一个 Bank 类(用于存放账户金额，并提供柜台取钱和 ATM 机取钱两种方法)、一个张三类（代表在柜台取钱）、一个李四类（代表在 ATM 机取钱）；

Bank 类中提供的两种取钱方法，方法内部对账户金额进行操作前有加锁操作 self.lock.acquire()，操作后有解锁操作 self.lock.release()；

Bank 类构造时初始化了账户金额，并使用 self.lock = threading.Lock()创建了线程同步锁；

两种取钱方法对同一账户进行取钱操作，故创建同一个静态值作为账户金额；

张三类的构造函数和李四类的构造函数传入的是同一个 Bank 类的对象，说明它们操作的是同一账户；

张三类和李四类均继承 threading.Thread 类，重写了其 run()方法，循环执行取钱操作；

创建 2 个线程模拟张三和李四的同时取款，start()启动线程。

七、程序代码

题目一:

```
import threading
import time


class BoxOffice(threading.Thread):
    def __init__(self, num):
        threading.Thread.__init__(self)
        self.setName('Window ' + str(num))  # 售票窗口号

    def run(self):
        global ticket
        global lock

        while ticket > 0:
            lock.acquire()  # 加锁
            print(self.getName() + ' sells ticket ' + str(ticket))
            ticket = ticket - 1  # 票数减 1
            lock.release()  # 解锁
            time.sleep(0.1)  # 每次售票完毕后休眠 100ms


if __name__ == '__main__':
    ticket = 30  # 共 30 张电影票

    lock = threading.Lock()  # 创建锁

    threads = []
    for i in range(4):  # 创建 4 个线程
        threads.append(BoxOffice(i + 1))

    for t in threads:
        t.start()
```

题目二:

```
import threading
import time

class Bank:
    def __init__(self):
        self.money = 5000  # 初始 5000 元
        self.lock = threading.Lock()  # 创建锁
```

```
    def counter_withdraw(self, money1):
        self.lock.acquire()  # 加锁
        self.money = self.money - money1
        print('ZhangSan gets $' + str(money1) + ' from counter, and
remains $' + str(self.money) + ' in the account')
        self.lock.release()  # 解锁

    def atm_withdraw(self, money2):
        self.lock.acquire()  # 加锁
        self.money = self.money - money2
        print('LiSi gets $' + str(money2) + ' from ATM, and remains
$' + str(self.money) + ' in the account')
        self.lock.release()  # 解锁

class ZhangSan(threading.Thread):
    def __init__(self, same_account):  # 初始化为同一账户
        threading.Thread.__init__(self)
        self.bank = same_account

    def run(self):
        while self.bank.money >= 100:
            self.bank.counter_withdraw(100)  # 柜台取钱 100
            time.sleep(0.05)

class LiSi(threading.Thread):
    def __init__(self, same_account):  # 初始化为同一账户
        threading.Thread.__init__(self)
        self.bank = same_account

    def run(self):
        while self.bank.money >= 300:
            self.bank.atm_withdraw(300)  # ATM 取钱 300
            time.sleep(0.1)


if __name__ == '__main__':
    account = Bank()

    p1 = ZhangSan(account)
    p2 = LiSi(account)
    p1.start()
    p2.start()
```

八、实验结果

```
"D:\Kent's Workspace\java\Parallel_
Window 1 sells ticket 30
Window 2 sells ticket 29
Window 3 sells ticket 28
Window 4 sells ticket 27
Window 3 sells ticket 26
Window 2 sells ticket 25
Window 1 sells ticket 24
Window 4 sells ticket 23
Window 4 sells ticket 22
Window 3 sells ticket 21
Window 1 sells ticket 20
Window 2 sells ticket 19
Window 1 sells ticket 18
Window 3 sells ticket 17
Window 2 sells ticket 16
Window 4 sells ticket 15
Window 1 sells ticket 14
Window 4 sells ticket 13
Window 2 sells ticket 12
Window 3 sells ticket 11
Window 1 sells ticket 10
Window 2 sells ticket 9
Window 3 sells ticket 8
Window 4 sells ticket 7
Window 1 sells ticket 6
Window 2 sells ticket 5
Window 4 sells ticket 4
Window 3 sells ticket 3
Window 1 sells ticket 2
Window 2 sells ticket 1

Process finished with exit code 0
```

从程序运行结果可知，4 个售票窗口线程同时卖票，不同窗口卖出不同的票，没有出现卖出同一张票的情况。

题目二:

```
"D:\Kent's Workspace\java\Parallel_Computing\4th\venv\Scripts\python.
ZhangSan gets $100 from counter, and remains $4900 in the account
LiSi gets $300 from ATM, and remains $4600 in the account
ZhangSan gets $100 from counter, and remains $4500 in the account
ZhangSan gets $100 from counter, and remains $4400 in the account
LiSi gets $300 from ATM, and remains $4100 in the account
ZhangSan gets $100 from counter, and remains $4000 in the account
ZhangSan gets $100 from counter, and remains $3900 in the account
LiSi gets $300 from ATM, and remains $3600 in the account
ZhangSan gets $100 from counter, and remains $3500 in the account
LiSi gets $300 from ATM, and remains $3200 in the account
ZhangSan gets $100 from counter, and remains $3100 in the account
ZhangSan gets $100 from counter, and remains $3000 in the account
LiSi gets $300 from ATM, and remains $2700 in the account
ZhangSan gets $100 from counter, and remains $2600 in the account
ZhangSan gets $100 from counter, and remains $2500 in the account
LiSi gets $300 from ATM, and remains $2200 in the account
ZhangSan gets $100 from counter, and remains $2100 in the account
ZhangSan gets $100 from counter, and remains $2000 in the account
LiSi gets $300 from ATM, and remains $1700 in the account
ZhangSan gets $100 from counter, and remains $1600 in the account
ZhangSan gets $100 from counter, and remains $1500 in the account
LiSi gets $300 from ATM, and remains $1200 in the account
ZhangSan gets $100 from counter, and remains $1100 in the account
ZhangSan gets $100 from counter, and remains $1000 in the account
LiSi gets $300 from ATM, and remains $700 in the account
ZhangSan gets $100 from counter, and remains $600 in the account
ZhangSan gets $100 from counter, and remains $500 in the account
LiSi gets $300 from ATM, and remains $200 in the account
ZhangSan gets $100 from counter, and remains $100 in the account
ZhangSan gets $100 from counter, and remains $0 in the account

Process finished with exit code 0
```

从程序运行结果可知，张三和李四同时调用 Bank 类的不同方法对同一账户进行取款，由于取款过程有同步锁，无法多个线程同时对账户金额进行更改操作，因此没有出现账户金额的一致性问题。

九、总结建议

经过本次实验，我对并发程序设计的原理加深了认识；对并发程序设计时需要注意的同步问题、一致性问题（如 threading.Lock()创建锁）有了进一步了解；同时更加熟悉 Python 语言在多线程编程方面的编程模式(如继承 threading.Thread 并重写 run()方法)，进而提高了个人的 Python 编程能力。