بسم تعالی

سیستم‌های عامل

تمرین چهارم

استاد:

دکتر حسین اسدی

نویسنده :

محمدهومان کشوری

شماره دانشجویی :

99105667

تمرینات عملی

سوال 2.

**** تمامی کدها به این نوشته ضمیمه شده‌اند****

ابتدا کد برای اجرای تردها را می‌نویسیم

```
from threading import Thread
from time import sleep, time
import requests
def task():
    # block for a moment
    r = requests.get('http://example.org')
    # display a message
    # print(r)
if __name__ == '__main__':
    num_of_threads = 20
    num_of_iterations = 1
    times = []
    for i in range(num_of_iterations):
        t = time()
        threads = []
        for i in range(num_of_threads):
            thread = Thread(target=task)
            threads.append(thread)
            thread.start()
        for thread in threads:
            thread.join()
        t = time() - t
        # print(t)
        times.append(t)
        # sleep(1)
    avg_time = sum(times) / len(times)
    print(avg_time)
```

در کد بالا 20 ترد داریم که به دامنه خواسته شده ریکوئست می‌زنند و سپس جواب را به ما برمی‌گردانند.

در قسمت دوم :

یک کد تستر(tester.py) می‌نویسیم که به تعداد دلخواه دو کد ما را اجرا کرده و نیز درنهایت از هر دو میانگین گرفته و خروجی را به ما نشان می‌دهد(کد زیر)

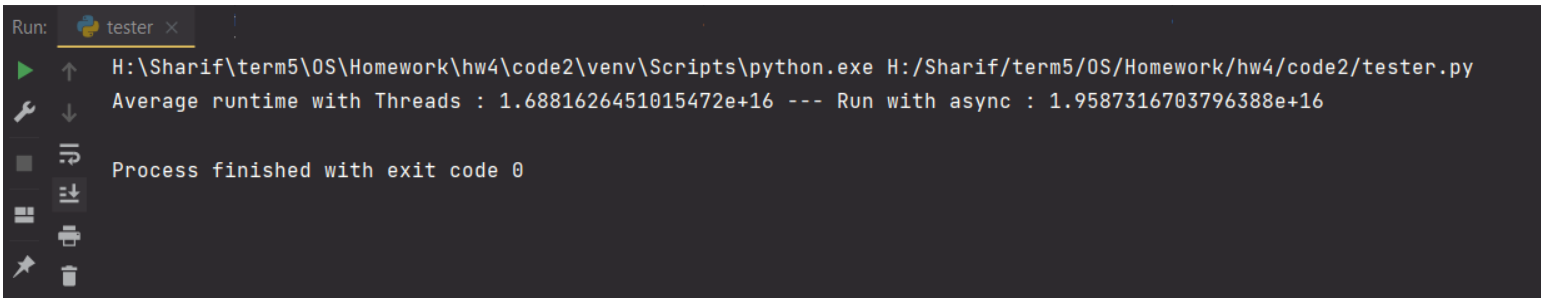
```
import re
import subprocess
import sys
from subprocess import run, PIPE
from time import sleep


def get_test(num_of_test):
    t1 = []
    t2 = []
    for i in range(num_of_test):
        # p1 = run([sys.executable + "/main.py"], stdout=PIPE,
        #          # input=f"""{stri}""",
        #          encoding='ascii')\
        executable_path =
'H:\\Sharif\\term5\\OS\\Homework\\hw4\\code2\\venv\\Scripts\\python.exe'
        # sleep(0.5)
        out2 = subprocess.check_output([executable_path, './main2.py'])
        out2 = float(str(re.sub("[^0-9]", "", out2.decode("utf-8") )))
        # sleep(0.5)
        out1 = subprocess.check_output([executable_path, './main.py'])
        out1 = float(str(re.sub("[^0-9]", "", out1.decode("utf-8"))))
        # list(p.stdout.split(" "))

        t1.append(out1)
        t2.append(out2)

    print(f"Average runtime with Threads : {sum(t1)/num_of_test} --- Run
with async : {sum(t2)/num_of_test}")
if __name__ == '__main__':
    get_test(20)
```

متغیر executable path مکان پایتون اجرا کننده دو کد را گرفته و با استفاده از subprocess هر دو را اجرا کرده و در خروجی out1 و out2 می‌ریزد.
حال 20 مرتبه هر دو کد را اجرا کرده و سپس میانگین را در انتها نشان می‌دهیم.
جواب نهایی:



پس مشخص می‌شود زمان اجرا با ترد **کمتر** بوده.
در ابتدای امر توجه کنیم که در برنامه async (در برنامه ما main2.py) درخواست‌ها را ارسال می‌کنیم اما منتظر جواب request.get نمی‌مانیم و ادامه کد را اجرا می‌کنیم و await منتظر جواب آن می‌ماند در صورتی که در برنامه multi-thread (که در برنامه ما main.py است) همزمان 20 مرتبه به دامنه درخواست فرستاده و منتظر جواب می‌ماند.
حال نکته این است که در برنامه async، درست است درخواست‌ها بدون صبر برای جواب ارسال می‌شوند اما گرفتن جواب‌های آنها موقع گرفتن جواب از example.org زمان زیادی می‌برد چون به ترتیب جواب‌ها گرفته می‌شوند.
اینطور به مسئله نگاه کنید که انگار یک فرد (در برنامه ما ترد اصلی) 20 بار به یک سایت درخواست می‌زند (بدون توجه به جواب درخواست) اما موقع گرفتن جواب باید تک به تک جواب‌ها را بررسی کند.
اما در برنامه multi-thread درخواست‌ها به صورت موازی زده شده و جواب‌ها نیز به صورت موازی گرفته می‌شود مانند این که چند فرد همزمان به یک دامنه درخواست ارسال کنند و به صورت موازی نیز جواب دریافتی را بررسی کنند.