

Tugas Pendahuluan #: THREAD ¹

Sistem Terdistribusi dan Parallel

December 16, 2022

Nama :FarizAbqariFawwazIllahi

NIM :1301204063

Kelas : IF 44 06

Topik: Thread (20)

1. Jelaskan bagaimana thread bekerja! Sertakan contoh konkrit permasalahan yang dapat dieksekusi dengan parallel

Thread adalah sebuah unit terkecil dari proses yang dapat dieksekusi secara parallel pada sistem operasi. Setiap proses pada sistem operasi biasanya memiliki setidaknya satu thread yang disebut thread utama, tetapi sebuah proses juga dapat memiliki beberapa thread tambahan yang disebut thread sekunder.

Thread bekerja dengan cara membagi tugas-tugas yang harus diselesaikan oleh sebuah proses menjadi beberapa bagian kecil yang dapat dieksekusi secara parallel. Dengan demikian, thread dapat membantu mempercepat proses penyelesaian suatu tugas dengan cara mengelola dan mengeksekusi bagian-bagian tersebut secara bersamaan.

Sebagai contoh, jika sebuah proses harus menghitung jumlah dari sekumpulan data yang sangat besar, maka proses tersebut dapat membuat beberapa thread untuk mengelola bagian-bagian dari data tersebut dan mengeksekusi pengolahan data tersebut secara parallel. Dengan demikian, proses tersebut akan dapat menyelesaikan tugasnya lebih cepat daripada jika hanya menggunakan satu thread saja.

Namun, perlu diingat bahwa penggunaan thread juga dapat menyebabkan beberapa masalah seperti konflik akses ke sumber daya yang sama atau masalah sinkronisasi antar thread. Oleh

¹ Squanix

karena itu, penggunaan thread harus dilakukan dengan hati-hati dan terkontrol agar tidak menimbulkan masalah yang tidak diinginkan.

2. Jelaskan siklus hidup thread! **Topik: Pemrograman Thread(80)**

Siklus hidup thread adalah serangkaian tahapan yang dialami oleh sebuah thread sejak dibuat hingga dihentikan atau diakhiri. Berikut ini adalah beberapa tahapan dalam siklus hidup thread:

1. **New:** Tahap ini adalah tahap dimana sebuah thread baru dibuat. Pada tahap ini, thread belum siap untuk dieksekusi dan masih dalam keadaan tidak aktif.
2. **Runnable:** Tahap ini adalah tahap dimana sebuah thread siap untuk dieksekusi oleh sistem operasi. Pada tahap ini, thread akan menunggu gilirannya untuk dieksekusi sesuai dengan mekanisme scheduling yang digunakan oleh sistem operasi.
3. **Running:** Tahap ini adalah tahap dimana sebuah thread sedang dieksekusi oleh sistem operasi. Pada tahap ini, thread akan menjalankan tugas-tugas yang telah diberikan kepadanya sampai selesai atau sampai mencapai tahap selanjutnya.
4. **Waiting:** Tahap ini adalah tahap dimana sebuah thread sedang menunggu suatu kondisi atau kejadian tertentu terjadi sebelum dapat melanjutkan eksekusinya. Pada tahap ini, thread akan tetap dalam keadaan tidak aktif sampai kondisi atau kejadian yang ditunggu terjadi.
5. **Terminated:** Tahap ini adalah tahap dimana sebuah thread dihentikan atau diakhiri. Pada tahap ini, thread tidak dapat dieksekusi lagi dan tidak dapat diaktifkan kembali.

Siklus hidup thread akan terus berulang selama thread tersebut masih diperlukan dan tidak dihentikan atau diakhiri. Namun, siklus hidup thread juga dapat terputus pada setiap tahapnya, tergantung pada kebutuhan dan kondisi yang ada.

1. Buatlah program berbasis thread untuk mencari banyaknya bilangan prima pada rentang bilangan tertentu! Misalkan $n=20000$ dan $m=9999$. Anda diharapkan dapat menghitung banyaknya bilangan prima yang terdapat pada rentang n dan m .

```
thread1_tanpaQ.py - C:\Users\fariz\Downloads\TP_SISTER_MOD10_1301204063\thread1_tanpaQ.py (3.10.7)
File Edit Format Run Options Window Help

import threading

def is_prime(m):
    if m <= 1:
        return False
    for i in range(2, int(m ** 0.5) + 1):
        if m % i == 0:
            return False
    return True

def count_primes(n, m, results):
    count = 0
    for i in range(n, m+1):
        if is_prime(i):
            count += 1
    results.append(count) # Menambahkan hasil dari setiap thread ke list

def main():
    # Membagi rentang menjadi 4 bagian yang sama
    n = 20000
    m = 9999
    num_threads = 4
    step = (n - m + 1) // num_threads
    threads = []
    results = []
    for i in range(num_threads):
        start = m + i * step
        end = start + step - 1
        if i == num_threads - 1:
            end = n
        thread = threading.Thread(target=count_primes, args=(start, end, results))
        thread.start()
        threads.append(thread)

    # Menunggu semua thread selesai
    for thread in threads:
        thread.join()
    # Menambahkan hasil dari setiap thread ke list

    # Menambahkan hasil dari setiap thread
    total_count = sum(results)
    print(f'Jumlah bilangan prima: {total_count}')

if __name__ == '__main__':
    main()
```

```
IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help

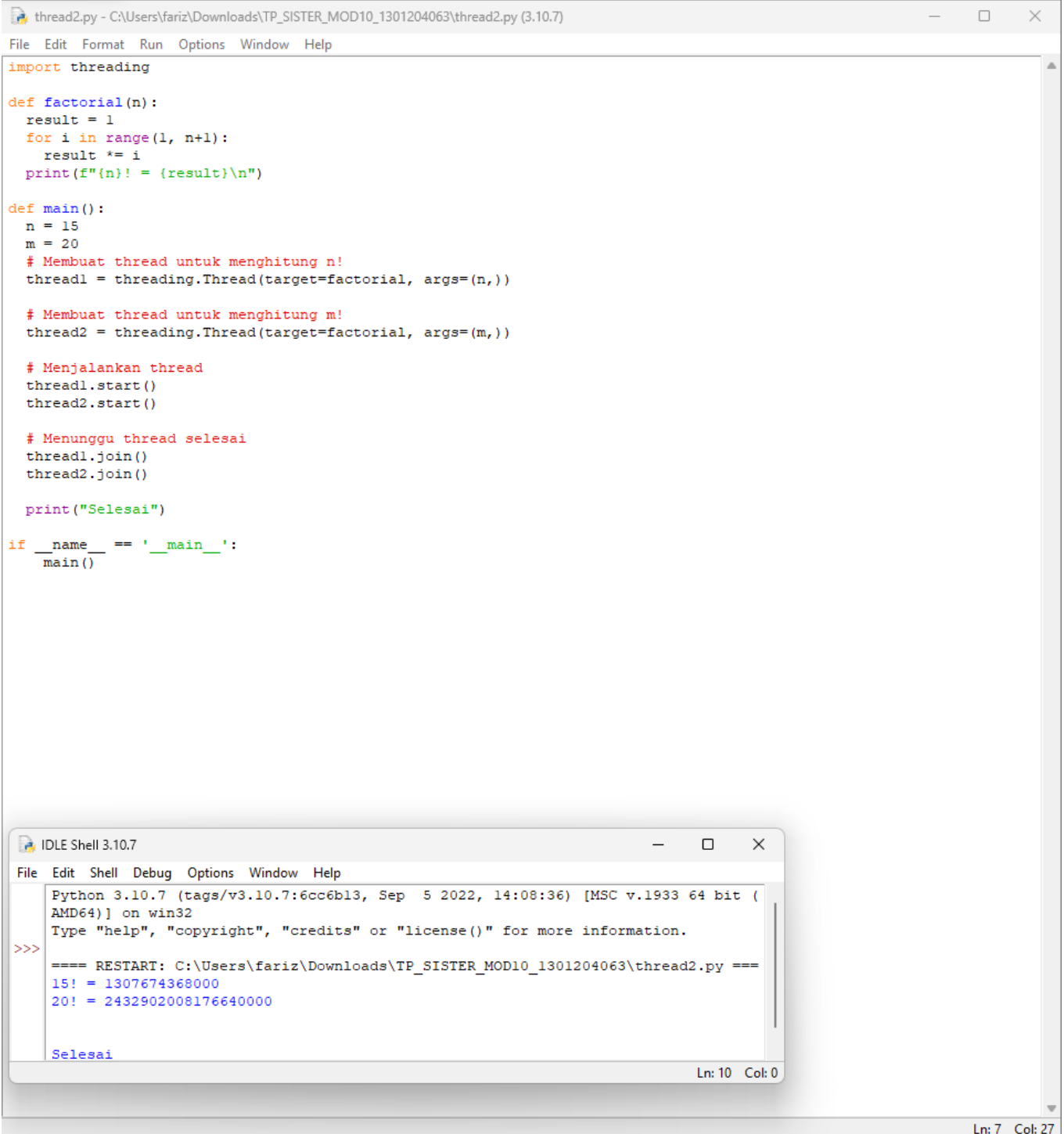
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: C:\Users\fariz\Downloads\TP_SISTER_MOD10_1301204063\thread1_tanpaQ.py
Jumlah bilangan prima: 1033
>>>
```

Ln: 6 Col: 0

Ln: 45 Col: 0

2. Buatlah program berbasis thread untuk melakukan operasi faktorial. Misalkan terdapat $n=15$ dan $m=20$. Maka Anda harus membuat program untuk menghitung $n!$ dan $m!$ secara parallel.



The image shows a Python IDE window titled 'thread2.py - C:\Users\fariz\Downloads\TP_SISTER_MOD10_1301204063\thread2.py (3.10.7)'. The code defines a `factorial` function and a `main` function that uses `threading.Thread` to calculate $15!$ and $20!$ in parallel. Below the code editor is an 'IDLE Shell 3.10.7' window showing the program's output.

```
import threading

def factorial(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    print(f"{n}! = {result}\n")

def main():
    n = 15
    m = 20
    # Membuat thread untuk menghitung n!
    thread1 = threading.Thread(target=factorial, args=(n,))

    # Membuat thread untuk menghitung m!
    thread2 = threading.Thread(target=factorial, args=(m,))

    # Menjalankan thread
    thread1.start()
    thread2.start()

    # Menunggu thread selesai
    thread1.join()
    thread2.join()

    print("Selesai")

if __name__ == '__main__':
    main()
```

The shell output shows the program's execution details and results:

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\fariz\Downloads\TP_SISTER_MOD10_1301204063\thread2.py ====
15! = 1307674368000
20! = 2432902008176640000

Selesai
```

Ln: 10 Col: 0

Ln: 7 Col: 27