

Andry Syva Maldini  
21083010085/Sistem Operasi Kelas-A

## Laporan Tugas 8

### A. Script dari soal Latihan multiprocessing.

```
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

x = int(input("Batas perulangan: "))
def tampil(i):
    if i % 2 == 0:
        print(f'{i+1} Genap', "- ID proses", getpid())
    elif i % 2 != 0:
        print(f'{i+1} Ganjil', "- ID proses", getpid())
    else:
        print("error")
    sleep(1)

print("\nSekuensial")

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
sekuensial_awal = time()

# PROSES BERLANGSUNG
for i in range(x):
    tampil(i)

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()

print("\nKelas Process")

# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()

# PROSES BERLANGSUNG
for i in range(x):
    p = Process(target=tampil, args=(i,))
    kumpulan_proses.append(p)
    p.start()

# UNTUK MENGGABUNGKAN PROSES-PROSES AGAR TIDAK LONCAT KE
# PROSES SEBELUM'NYA
for i in kumpulan_proses:
```

```

p.join()

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
process_akhir = time()

print("\nKelas Pool")

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(tampil, range(x))
pool.close()

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()

# Hasil Akhir
print("\nSekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")

```

## B. Penjelasan mengenai script

1. Mengimport library yang akan digunakan untuk menjalankan script pada python.

```

from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

```

2. Fungsi ini digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Kita panggil fungsi sleep untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan.

```

x = int(input("Batas perulangan: "))
def tampil(i):
    if i % 2 == 0:
        print(f"{i+1} Ganjil", "- ID proses", getpid())
    elif i % 2 != 0:
        print(f"{i+1} Genap", "- ID proses", getpid())
    else:
        print("error")
    sleep(1)

```

3. Pemrosesan Sekuensial

```

print("\nSekuensial")

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
sekuensial_awal = time()

# PROSES BERLANGSUNG
for i in range(x):
    tampil(i)

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()

```

#### 4. Multiprocessing dengan kelas Process

Dapat diperhatikan dengan seksama bahwa ID proses tiap memanggil fungsi cetak adalah berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain. Kumpulan proses harus ditampung dan digabung menjadi satu (`p.join()`) agar tidak merambah ke proses selanjutnya. Silahkan eksekusi file berikut pada terminal anda, maka anda akan paham apa yang saya maksudkan.

```

# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()

# PROSES BERLANGSUNG
for i in range(x):
    p = Process(target=tampil, args=(i,))
    kumpulan_proses.append(p)
    p.start()

# UNTUK MENGABUNGKAN PROSES-PROSES AGAR TIDAK LONCAT KE PROSES SEBELUM'NYA
for i in kumpulan_proses:
    p.join()

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
process_akhir = time()

```

#### 5. Multiprocess dengan kelas Pool

Jumlah ID proses terbatas pada empat saja karena jumlah CPU pada komputer saya hanyalah 6. Jangan risaukan urutan angka yang dicetak jika tidak berurutan, kan memang ini pemrosesan paralel. Fungsi `map()` itu memetakan pemanggilan fungsi cetak ke dalam 6 CPU sebanyak 10 kali.

```

print("\nKelas Pool")

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(tampil, range(x))
pool.close()

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()

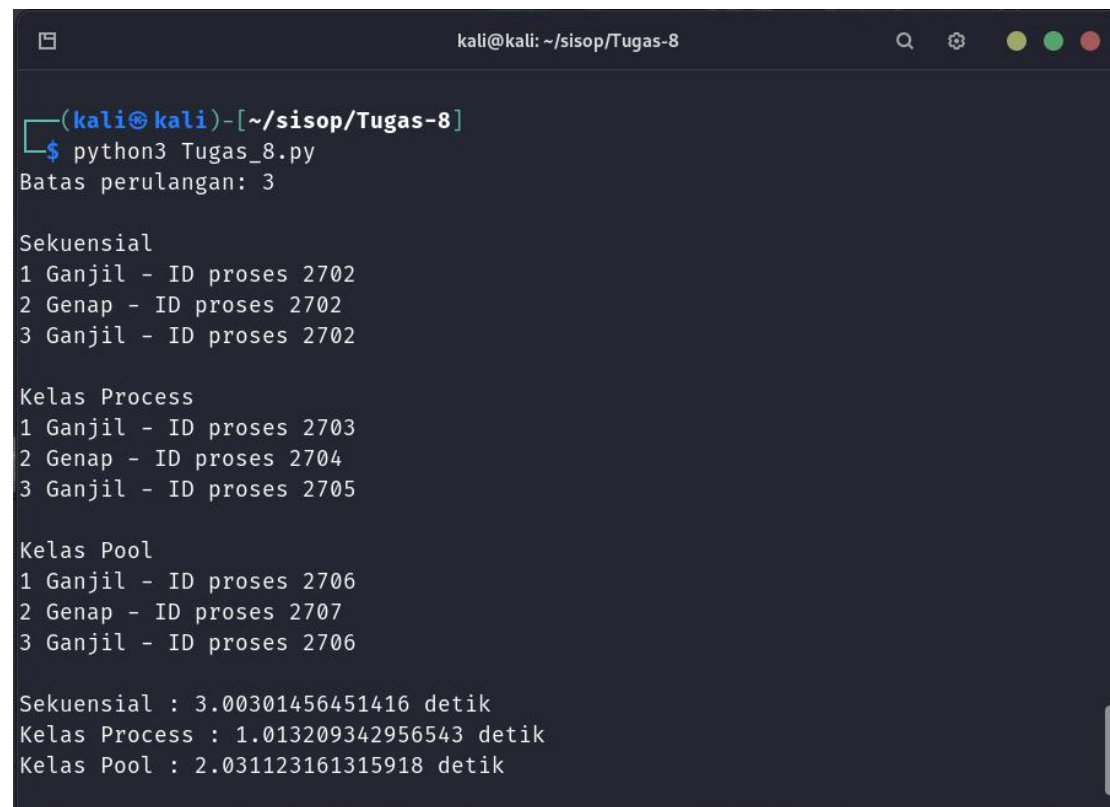
```

## 6. Bandingkan Waktu Eksekusi

Sudah sewajarnya proses sekuensial lebih lambat dibanding multiprocessing namun bukan berarti kita harus melakukan multiprocessing terus menerus, gunakan metode sesuai kebutuhan.

```
# Hasil Akhir
print("\nSekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")
```

## C. Menjalan Scirpt multiprocessing.



```
kali@kali: ~/sisop/Tugas-8

(kali@kali)-[~/sisop/Tugas-8]
$ python3 Tugas_8.py
Batas perulangan: 3

Sekuensial
1 Ganjil - ID proses 2702
2 Genap - ID proses 2702
3 Ganjil - ID proses 2702

Kelas Process
1 Ganjil - ID proses 2703
2 Genap - ID proses 2704
3 Ganjil - ID proses 2705

Kelas Pool
1 Ganjil - ID proses 2706
2 Genap - ID proses 2707
3 Ganjil - ID proses 2706

Sekuensial : 3.00301456451416 detik
Kelas Process : 1.013209342956543 detik
Kelas Pool : 2.031123161315918 detik
```