Nama : Muhammad Daud Rajasa

Npm : 21083010060

Sistem Operasi B

A. Script dari soal Latihan multiprocessing

```
  GNU nano 6.2                                    tugas8.py
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} Ganjil", "- ID proses", getpid())
  elif i % 2 !=0:
    print(f"{i+1} Genap", "- ID proses", getpid())
  else:
    print("selesai")
    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()

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

Pengertian dari script 1.

Kita import dulu library yang diperlukan

```
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("selesai")
    sleep(1)
```

3. Pemrosesan Sekuensial **Pemrosesan sekuensial** adalah **pemrosesan** secara satu-persatu, dari sekumpulan informasi sejenis yang setiap elemennya dapat diakses dengan keterurutan tertentu (ada suksesor), jadi seakan-akan kumpulan elemen merupakan "Deret" elemen.

```
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

```
# 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()
```

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 selanjut'nya 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.

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 emang 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. Waktu Eksekusi

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

Sudah sewajarnya proses sekuensial lebih lambat dibanding multiprocessing namun bukan berarti kita harus melakukan multiprocessing terus menerus, gunakan metode sesuai kebutuhan. Nah apabila barisan kode di atas dikumpulkan jadi satu maka jadinya akan seperti ini.

Untuk outputnya sperti dibawah

```
muhammad@muhammad-VirtualBox:~/tugas8$ python3 tugas8.py
Batas perulangan: 3

Sekuensial
1 Ganjil - ID proses 6563
2 Genap - ID proses 6563
3 Ganjil - ID proses 6563

Kelas Process
1 Ganjil - ID proses 6682
2 Genap - ID proses 6683
3 Ganjil - ID proses 6684

Kelas Pool
1 Ganjil - ID proses 6685
2 Genap - ID proses 6685
3 Ganjil - ID proses 6685

Sekuensial : 2.0742416381835938e-05 detik
Kelas Process : 0.01056981086730957 detik
Kelas Pool : 0.043297529220581055 detik
muhammad@muhammad-VirtualBox:~/tugas8$
```