Nama : Mareta Putri Wardhana
NPM : 21083010002
Kelas : Sistem Operasi B

Latihan Multiprocessing

Soal :
Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap!
Batasan :

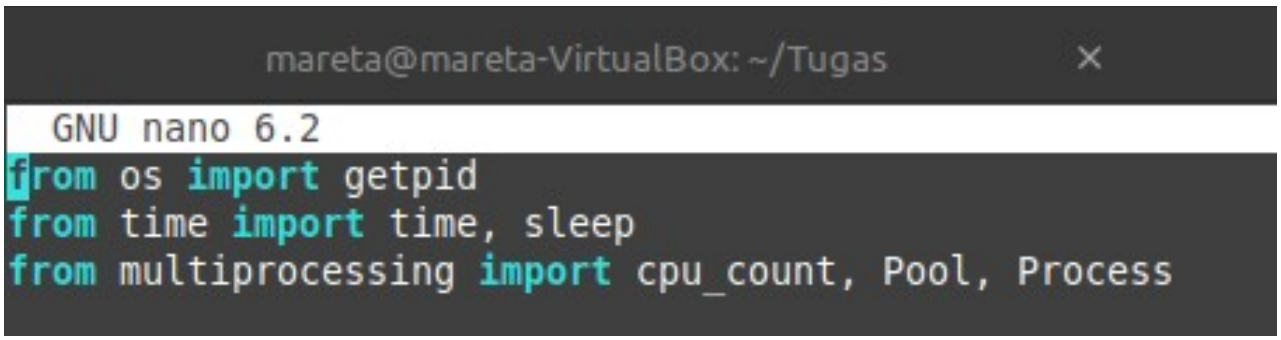
- Nilai yang dijadikan argument pada fungsi sleep() adalah satu detik.
- Masukkan jumlahnya satu dan berupa bilangan bulat.
- Masukkan adalah batas dari perulangan tersebut.
- Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan paralel.

Penyelesaian :
Membuat file dengan perintah nano Tugas_8.py

```
mareta@mareta-VirtualBox:~/Tugas$ nano Tugas_8.py
```

Selanjutnya menuliskan script yang kita inginkan.
Pertama-tama, kita import terlebih dahulu built-in libraries yang akan kita gunakan yaitu :

- getpid digunakan untuk mengambil ID proses.
- time digunakan untuk mengambil waktu (detik).
- sleep digunakan untuk memberi jeda waktu (detik).
- cpu_count digunakan untuk melihat jumlah CPU.
- Pool digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer.
- Process digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.



Kedua, inisialisasikan fungsi. Fungsi ini digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Pada fungsi ini terdapat conditional statements if...else untuk menentukan apakah sebuah bilangan ganjil atau genap. Kemudian, memanggil fungsi sleep untuk memberi jeda waktu selama 1 detik. Selanjutnya, inisialisasikan n sebagai inputan range angka dengan tipe integer.

```
def cetak(i):
    if (i+1)%2==0:
       print(i+1, "Genap - ID proses", getpid())
       sleep(1)
    else:
       print(i+1, "Ganjil - ID proses", getpid())
       sleep(1)

n = int(input("Masukkan Range Angka : "))
```

Ketiga, melakukan proses sekuensial. Gunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Lalu, loop for...in digunakan untuk memproses sekuensial sebanyak range n. Pada fungsi loop ini masukkan fungsi cetak yang telah diinisialisasikan sebelumnya.

```
print("Sekuensial")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
sekuensial_awal = time()

# PROSES BERLANGSUNG
for i in range(n):
    cetak(i)

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()
```

Keempat, melakukan multiprocessing dengan kelas Process. Gunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Inisialisasikan array kosong untuk menampung kumpulan proses. Lalu, loop for...in digunakan untuk memproses multiprocessing dengan kelas Process sebanyak range n. Pada loop ini masukkan fungsi cetak yang telah diinisialisasikan sebelumnya, tetapi dengan ID proses yang berbeda-beda. Hal ini berarti tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain. Kumpulan proses ditampung dan digabung menjadi satu pada p.join().

```
print("multiprocessing.Process")
# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()

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

# UNTUK MENGGABUNGKAN PROSES-PROSES AGAR TIDAK LONCAT KE PROSES SEBELUMNYA
for i in kumpulan_proses:
    p.join()

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
process_akhir = time()
```

Kelima, melakukan multiprocessing dengan kelas Pool. Gunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Fungsi map() digunakan untuk memetakan pemanggilan fungsi cetak ke dalam CPU yang dimiliki komputer sebanyak n kali. Jumlah ID proses terbatas tergantung jumlah CPU pada komputer. Untuk urutan angka yang dicetak dapat tidak terurut karena ini merupakan pemrosesan paralel.

```
print("multiprocessing.Pool")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(0,n))
pool.close

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()
```

Keenam, tampilkan waktu eksekusi pemrosesan sekuensial dan paralel dengan mengurangi waktu sesudah dengan waktu sebelum eksekusi.

```
print("Waktu eksekusi sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", process_akhir - process_awal, "detik")
print("Waktu eksekusi multiprocessing.Pool :", pool_akhir - pool_awal, "detik")
```

Jika sudah selesai menuliskan script, klik ctrl+x, kemudian y dan enter.
Untuk melihat hasil output kita dapat menggunakan perintah python3 Tugas_8.py. Kemudian, kita masukkan range angka yaitu 3. Sehingga, hasilnya pada batas dari perulangan pemrosesan sekuensial dan paralel adalah sebanyak 3 dan pada tiga baris terakhir ditunjukkan waktu dari setiap eksekusinya.

```
mareta@mareta-VirtualBox:~/Tugas$ python3 Tugas_8.py
Masukkan Range Angka : 3
Sekuensial
1 Ganjil - ID proses 6017
2 Genap - ID proses 6017
3 Ganjil - ID proses 6017
multiprocessing.Process
1 Ganjil - ID proses 6130
2 Genap - ID proses 6131
3 Ganjil - ID proses 6132
multiprocessing.Pool
1 Ganjil - ID proses 6133
2 Genap - ID proses 6133
3 Ganjil - ID proses 6133
Waktu eksekusi sekuensial : 3.0033750534057617 detik
Waktu eksekusi multiprocessing.Process : 1.1134414672851562 detik
Waktu eksekusi multiprocessing.Pool : 3.7393810749053955 detik
mareta@mareta-VirtualBox:~/Tugas$
```