Nama : Sandria Amelia Putri

NPM : 21083010005

Kelas : A

Soal Latihan Multiprocessing

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.**

Gunakan command **nano** untuk membuat file bash **Tugas_8.py**.

```
sandria@sandria-VirtualBox:~/Documents/SistemOperasi/Tugas8$ nano Tugas_8.py
```

Pertama, import terlebih dahulu built-in libraries yang akan digunakan. **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.

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

Kedua, inisialisasikan fungsi yang akan digunakan. 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. Selanjutnya, fungsi **sleep** dipanggil untuk memberi jeda waktu selama 1 detik.

```
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)
```

Ketiga, inisialisasikan **n** sebagai inputan range angka dengan tipe integer.

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

Keempat, lakukan pemrosesan **sekuensial**. Gunakan fungsi **time** untuk mendapatkan waktu sebelum dan sesudah eksekusi. Lalu, loop **for…in** digunakan untuk memproses sekuensial sebanyak range **n**. Pada 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()
```

Kelima, lakukan **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()** agar tidak merambah ke proses selanjutnya.

```
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()
```

Keenam, lakukan **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()
```

Ketujuh, 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")
```

Berikut merupakan hasil output dari script python di atas. Saya memasukkan range angka 3. Sehingga, batas dari perulangan pemrosesan sekuensial dan paralel adalah sebanyak 3. Pada tiga baris terakhir ditunjukkan waktu dari setiap eksekusinya. Proses sekuensial memang sudah sewajarnya lebih lambat dibanding multiprocessing. Namun, bukan berarti multiprocessing selalu digunakan terus menerus. Pakailah metode sesuai kebutuhan yang diperlukan.

```
sandria@sandria-VirtualBox:~/Documents/SistemOperasi/Tugas8$ python3 Tugas_8.py
Masukkan Range Angka : 3
Sekuensial
1 Ganjil - ID proses 1721
2 Genap - ID proses 1721
3 Ganjil - ID proses 1721
multiprocessing.Process
1 Ganjil - ID proses 1722
2 Genap - ID proses 1723
3 Ganjil - ID proses 1724
multiprocessing.Pool
1 Ganjil - ID proses 1725
2 Genap - ID proses 1725
3 Ganjil - ID proses 1725
Waktu eksekusi sekuensial : 3.0038294792175293 detik
Waktu eksekusi multiprocessing.Process : 1.0186481475830078 detik
Waktu eksekusi multiprocessing.Pool : 3.0217437744140625 detik
```