Calvien Danny N

21083010040 – Sistem Operasi A

A. Script dari soal Latihan multiprocessing

calviendanny@calviendanny-VirtualBox: ~/tugasku/sisop

File Edit View Search Terminal Help

GNU nano 6.2 Tugas_8.py

```
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
input = int(input("Masukkan batas: "))
def cetak(i):
  if i % 2 == 0:
    print((i+1),"Angka ganjil  - ID", getpid())
  else:
    print((i+1),"Angka genap  - ID", getpid())
  sleep(1)

print("Kelas sekuensial")
sekuensial_awal = time()
# PROSES BERLANGSUNG
for i in range(input):
    cetak(i)
# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()
print("\n Kelas Procces")
# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()
# PROSES BERLANGSUNG
for i in range(input):
    p = Process(target=cetak, 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("\n Kelas pool")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()
```

```
# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(0,input))
pool.close()
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()

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

B. 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
input = int(input("Masukkan batas: "))
```

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.

```
def cetak(i):
  if i % 2 == 0:
    print((i+1),"Angka ganjil  - ID", getpid())
  else:
    print((i+1),"Angka genap  - ID", getpid())
  sleep(1)
```

3. Pemrosesan Sekuensial

```
print("Kelas sekuensial")
sekuensial_awal = time()
# PROSES BERLANGSUNG
for i in range(input):
    cetak(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(input):
    p = Process(target=cetak, 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()
```

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

```
print("\n Kelas pool")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(0,input))
pool.close()
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()
```

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.

6. Bandingkan Waktu Eksekusi

```
print("\n Hasilnya")
print("Sekuensial :", 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.

Output

```
calviendanny@calviendanny-VirtualBox:~/tugasku/sisop$ python3 Tugas_8.py
Masukkan batas: 3
Kelas sekuensial
1 Angka ganjil  - ID 4868
2 Angka genap  - ID 4868
3 Angka ganjil  - ID 4868

 Kelas Procces
2 Angka genap  - ID 4870
3 Angka ganjil  - ID 4871
1 Angka ganjil  - ID 4869

 Kelas pool
1 Angka ganjil  - ID 4872
2 Angka genap  - ID 4872
3 Angka ganjil  - ID 4872

 Hasilnya
Sekuensial : 3.004255771636963 detik
Kelas Process : 1.011042594909668 detik
Kelas Pool : 3.023322105407715 detik
```