

Nama : Aulia Nur Fitriani

NPM : 21083010051

Kelas : Sistem Operasi B

Laporan Tugas 8 Multiprocessing

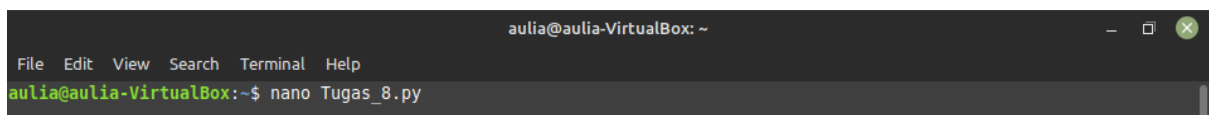
Soal Latihan 8:

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

Batasan :

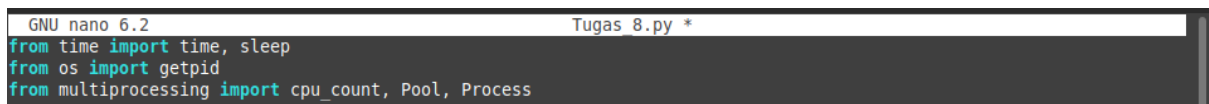
- Nilai yang dijadikan argument pada fungsi sleep() adalah satu detik
- Masukkan jumlahnya satu dan berupa bilangan bulat
- Masukkan adalah batas dari perulangan tersebut
- Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan paralel

Jawab:



```
aulia@aulia-VirtualBox: ~  
File Edit View Search Terminal Help  
aulia@aulia-VirtualBox:~$ nano Tugas_8.py
```

- Membuat file dengan nama file Tugas_8 dengan perintah nano



```
GNU nano 6.2 Tugas_8.py *  
from time import time, sleep  
from os import getpid  
from multiprocessing import cpu_count, Pool, Process
```

- Import library python yaitu dengan fungsi sebagai berikut
 - Time untuk mengambil waktu(detik)
 - Sleep untuk memberi jeda waktu(detik)
 - Getpid untuk mengambil ID proses
 - Cpu_count untuk melihat jumlah cpu
 - Pool adalah sebuah class pada library multiprocessing untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah cpu pada komputer
 - Process adalah sebuah class pada library multiprocessing untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer



```
print("input nilai :")  
nilai = int(input())  
  
def cetak(i):  
    for i in range(nilai):  
        if i % 2 == 0:  
            print(f"{i+1} Angka Ganjil", "- ID proses", getpid())  
        else:  
            print(f"{i+1} Angka Genap", "- ID proses", getpid())  
        sleep(1)  
    print(" ")
```

- Mendeklarasikan fungsi print dengan nama input nilai yang akan digunakan untuk memunculkan berapa jumlah nilai yang ingin dimasukkan.
- Def cetak untuk mendeklarasikan mana angka yang termasuk bilangan ganjil atau genap beserta ID proses sejumlah parameter yang digunakan dan disana saya menggunakan looping for dan fungsi if else. Kemudian saya menambahkan fungsi sleep(1) untuk menunda setiap satu detik untuk ke argumen selanjutnya.

```
# multiprocessing sekuensial
print("Multi_Sekuen")
sekuensial_awal = time()

for i in range(1):
    cetak(i)
sekuensial_akhir = time()
print(" ")
```

- Pada multiprocessing sekuensial dibuatlah sebuah pendeklarasian dan disini saya menggunakan looping for. Syntax “sekuensial_awal = time()” berfungsi untuk mendapatkan waktu sebelum di eksekusi lalu syntax for adalah berlangsungnya proses sekuensial. Dan range(1) artinya output yang akan dikeluarkan nanti hanya satu kali. Kemudian pada syntax “sekuensial_akhir = time()” untuk mendapatkan waktu setelah di eksekusi. Dan yang terakhir adalah memanggil syntax diatas dengan perintah print(“ “)

```
# multiprocessing process
print("Multi_Process")
kumpulan_proses = []
process_awal = time()

for i in range(1):
    p = Process(target = cetak, args = (i,))
    kumpulan_proses.append(p)
    p.start()

for i in kumpulan_proses:
    p.join()
process_akhir = time()
print(" ")
```

- Pada multiprocessing sekuensial dibuatlah sebuah pendeklarasian dan disini saya menggunakan looping for. Syntax “proses_awal = time()” berfungsi untuk mendapatkan waktu sebelum di eksekusi lalu syntax for adalah berlangsungnya proses sekuensial. Dan range(1) artinya output yang akan dikeluarkan nanti hanya satu kali. “for i in kumpulan_proses” digunakan untuk menggabungkan proses-proses agar tidak loncat ke proses sebelumnya. Kemudian pada syntax “proses_akhir = time()” untuk mendapatkan waktu setelah di eksekusi. Dan yang terakhir adalah memanggil syntax diatas dengan perintah print(“ “)

```
# multiprocessing pool
print("Multi_Pool")
pool_awal = time()
pool = Pool()
pool.map(cetak, range(0,1))
pool.close()
pool_akhir = time()
print(" ")
```

- Pada multiprocessing pool disini saya menggunakan fungsi .map() guna memetakan panggilan fungsi cetak kedalam sebanyak 1 kali

```
# membandingkan waktu 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")
```

- Terakhir kita akan membuat sebuah lama waktu pengekseskuan itu berlangsung, caranya yaitu dengan waktu akhir – waktu awal, sseperti yang telah kita deklarasikan pada setiap awal dan akhir dari sebuah program multiprocessing

```
aulia@aulia-VirtualBox:~$ python3 Tugas_8.py
input nilai :
3

Multi Sekuen
1 Angka Ganjil - ID proses 2208
2 Angka Genap - ID proses 2208
3 Angka Ganjil - ID proses 2208

Multi Process
1 Angka Ganjil - ID proses 2209
2 Angka Genap - ID proses 2209
3 Angka Ganjil - ID proses 2209

Multi Pool
1 Angka Ganjil - ID proses 2210
2 Angka Genap - ID proses 2210
3 Angka Ganjil - ID proses 2210

waktu eksekusi Sekuensial : 1.04506516456604 detik
waktu eksekusi Multiprocessing.Process : 1.012833595275879 detik
waktu eksekusi Multiprocessing.Pool : 1.0291719436645508 detik
```

- Output yang dihasilkan dari syntax diatas adalah sebagai berikut.