

Nama : Cesaria Deby Nurhalizah

NPM : 21083010120

Kelas : Sistem Operasi A

## **TUGAS 8 (MULTIPROCESSING)**

Multiprocessing adalah kemampuan sistem untuk menangani beberapa proses secara bersamaan dan independen. Dalam sistem multiprosesor, aplikasi dipecah menjadi rutinitas yang lebih kecil dan OS memberikan utas ke proses ini untuk kinerja yang lebih baik. Salah satu konsep dasar system operasi dengan multiprocessing adalah pemrograman paralel.

Pemrograman paralel adalah sebuah teknik eksekusi perintah yang mana dilakukan secara bersamaan pada CPU. Ada banyak kelas dalam modul multiprosesor python untuk membangun program paralel. Diantaranya tiga kelas dasar adalah Process, Queue, dan Lock yang akan membantu membangun program paralel. Menggunakan Python karena bahasa pemrograman tersebut sudah otomatis terinstal di hampir seluruh sistem operasi berbasis Linux selain itu secara default komputasi di Python dilakukan secara sekuensial.

### **SOAL LATIHAN :**

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

Batasan :

- Nilai yang dijadikan argumen 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.

```
cesariadby@cesariadby-VirtualBox: ~/Dokumen/Sistem_Operasi
Berkas  Sunting  Tampilan  Cari  Terminal  Tab  Bantuan
cesariadby@cesariad... x  cesariadby@cesariad... x  cesariadby@cesariad... x  cesariadby@cesariad... x  [icon] [dropdown]
GNU nano 6.2  Tugas_8.py
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

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

lim = int(input("Masukkan input: "))

#Sequential
sequential_first = time()
print("\n Sekuensial")
for i in range(lim):
    angka(i)
sequential_last = time()
```

### ***Built-in libraries:***

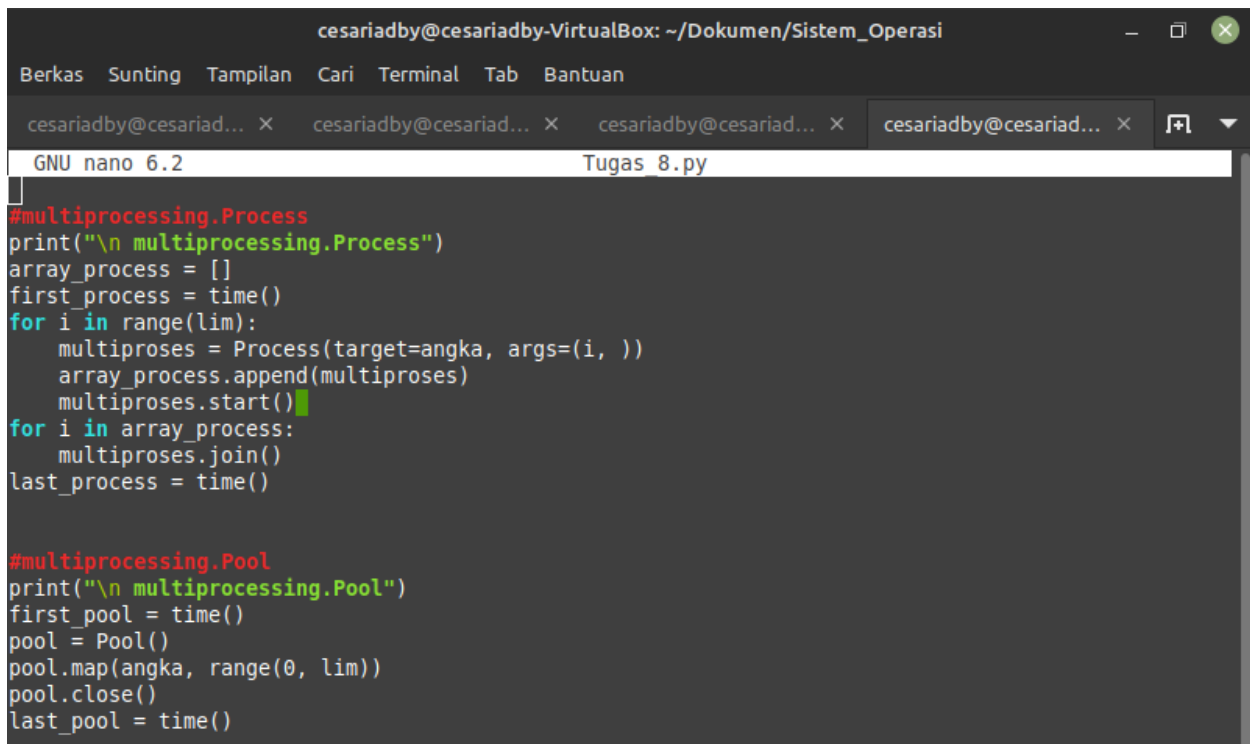
- **getpid** : mengambil ID proses.
- **time** : mengambil waktu(detik).
- **sleep** : memberi jeda waktu(detik).
- **cpu\_count** : melihat jumlah CPU.
- **Pool** : sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer.
- **Process**: sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.
- **Angka** : mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Menggunakan percabangan if-else untuk menjadikan variabel i yang terpanggil terbagi menjadi ganjil serta genap dengan ketentuan " $i \% 2 == 0$ ".

### ***1. Sekuensial***

Algoritma sekuensial (algoritma runtunan) merupakan salah satu struktur dasar algoritma yang bisa dikatakan cukup sederhana jika dibandingkan dengan struktur algoritma yang lain. Algoritma sekuensial bekerja dengan cara mengeksekusi setiap instruksi secara berurutan. Setiap instruksi akan dikerjakan satu per satu pada setiap barisnya dari awal hingga akhir, sesuai dengan urutan penulisan instruksi tersebut.

- Untuk mendapatkan waktu sebelum eksekusi  
sequential\_first = time()
- Proses berlangsung  
for i in range(lim):  
    angka(i)
- Untuk mendapatkan waktu setelah eksekusi  
sequential\_last = time()

Perulangan for dengan range disesuaikan pada inputan, kemudian menjalankan fungsi **angka** yang telah diinisialisasikan.



```

cesariadby@cesariadby-VirtualBox: ~/Dokumen/Sistem_Operasi
Berkas  Sunting  Tampilan  Cari  Terminal  Tab  Bantuan
cesariadby@cesariad... × cesariadby@cesariad... × cesariadby@cesariad... × cesariadby@cesariad... ×
GNU nano 6.2          Tugas 8.py

#multiprocessing.Process
print("\n multiprocessing.Process")
array_process = []
first_process = time()
for i in range(lim):
    multiproses = Process(target=angka, args=(i, ))
    array_process.append(multiproses)
    multiproses.start()
for i in array_process:
    multiproses.join()
last_process = time()

#multiprocessing.Pool
print("\n multiprocessing.Pool")
first_pool = time()
pool = Pool()
pool.map(angka, range(0, lim))
pool.close()
last_pool = time()

```

## 2. *Multiprocessing Process*

Perulangan for dengan range disesuaikan pada inputan, kemudian menjalankan fungsi **process** yang telah di-import dengan berdasarkan fungsi **angka**, dengan menambahkan

- Untuk menampung proses-proses  
array\_process = []

- Untuk mendapatkan waktu sebelum eksekusi  
first\_process = time()
- Proses berlangsung  
for I in range(lim):  
multiproses = Process(target=angka, args=(i, ))  
array\_process.append(multiproses)  
multiproses.start()
- Untuk menggabungkan proses-proses agar tidak loncat ke proses sebelumnya  
for i in array\_process:  
multiproses.join()
- Untuk mendapatkan waktu setelah eksekusi  
last\_process = time()

### 3. *Multiprocessing Pool*

Menggunakan fungsi map untuk memanggil fungsi **angka** ke dalam CPU sebanyak angka yang telah diinputkan.

- Untuk mendapatkan waktu sebelum eksekusi  
first\_pool = time()
- Proses berlangsung  
pool = Pool()  
pool.map(angka, range(0, lim))  
pool.close()
- Untuk mendapatkan waktu setelah eksekusi  
last\_pool = time()

### 4. *Perbandingan Waktu Eksekusi*

```
#Perbandingan.waktu
total_sequential = sequential_last - sequential_first
total_process = last_process - first_process
total_pool = last_pool - first_pool

print("\nWaktu eksekusi sequential: ", total_sequential, "detik")
print("Waktu eksekusi multiprocessing.Process: ", total_process, "detik")
print("Waktu eksekusi multiprocessing.Pool:" , total_pool, "detik")
□
```

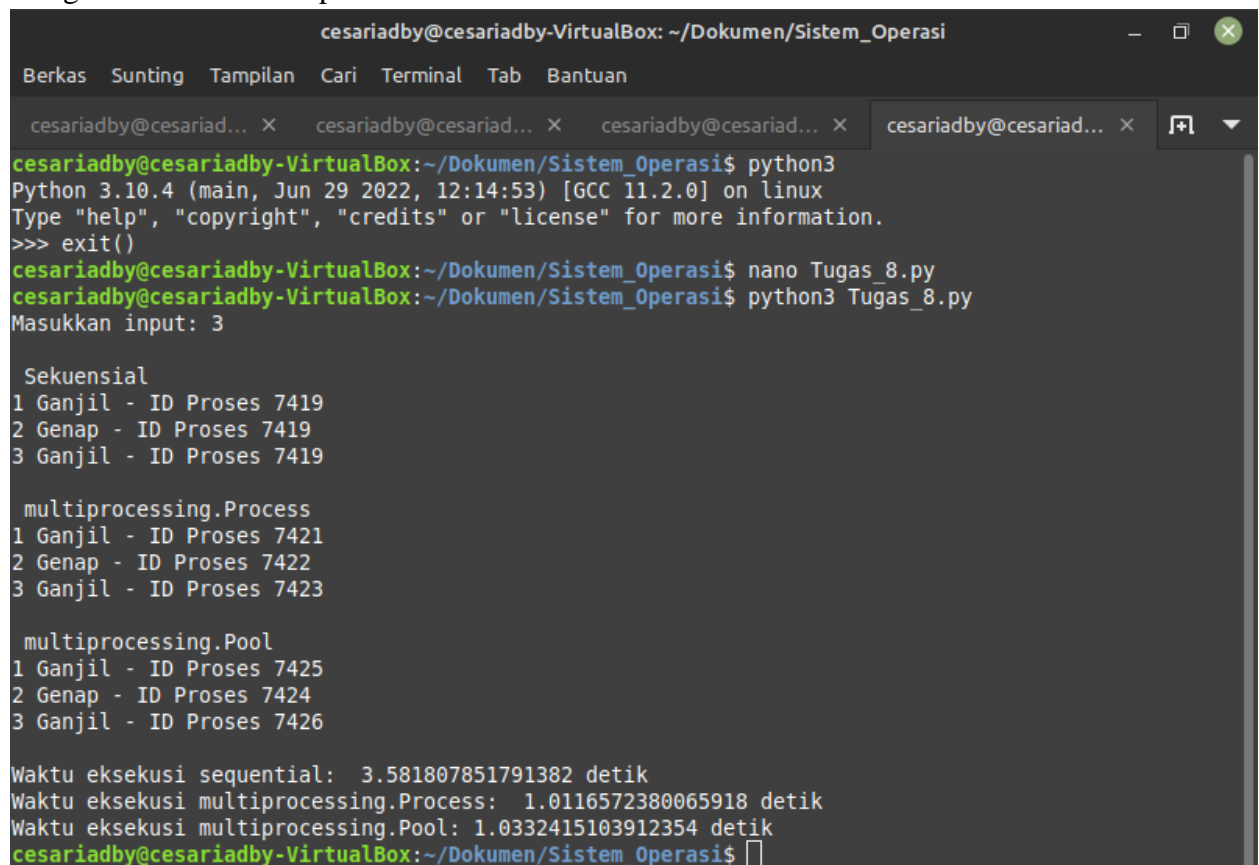
Setiap eksekusi dilakukan perhitungan dengan *last time* dikurangi dengan *first time*, sehingga akan didapatkan output sesuai dengan eksekusinya.

```
total_sequential = sequential_last - sequential_first
total_process = last_process - first_process
total_pool = last_pool - first_pool
```

- Selang waktu pada eksekusi Sekuensial  
`print("\nWaktu eksekusi sequential: ", total_sequential, "detik")`
- Selang waktu pada eksekusi Multiprocessing Process  
`print("Waktu eksekusi multiprocessing.Process: ", total_process, "detik")`
- Selang waktu pada eksekusi Multiprocessing Pool  
`print("Waktu eksekusi multiprocessing.Pool:" , total_pool, "detik")`

## Output :

Dengan memasukkan input : 3



```
cesariadby@cesariadby-VirtualBox: ~/Dokumen/Sistem_Operasi
Berkas  Sunting  Tampilan  Cari  Terminal  Tab  Bantuan
cesariadby@cesariad... X  cesariadby@cesariad... X  cesariadby@cesariad... X  cesariadby@cesariad... X  [?]
cesariadby@cesariadby-VirtualBox:~/Dokumen/Sistem_Operasi$ python3
Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
cesariadby@cesariadby-VirtualBox:~/Dokumen/Sistem_Operasi$ nano Tugas_8.py
cesariadby@cesariadby-VirtualBox:~/Dokumen/Sistem_Operasi$ python3 Tugas_8.py
Masukkan input: 3

Sekuensial
1 Ganjil - ID Proses 7419
2 Genap - ID Proses 7419
3 Ganjil - ID Proses 7419

multiprocessing.Process
1 Ganjil - ID Proses 7421
2 Genap - ID Proses 7422
3 Ganjil - ID Proses 7423

multiprocessing.Pool
1 Ganjil - ID Proses 7425
2 Genap - ID Proses 7424
3 Ganjil - ID Proses 7426

Waktu eksekusi sequential:  3.581807851791382 detik
Waktu eksekusi multiprocessing.Process:  1.0116572380065918 detik
Waktu eksekusi multiprocessing.Pool: 1.0332415103912354 detik
cesariadby@cesariadby-VirtualBox:~/Dokumen/Sistem_Operasi$
```