

# **OPERATION SYSTEM REPORT**

## **CLASS A**



### **“Tugas Minggu ke-8”**

#### **Name:**

Aniysah Fauziyyah Alfa (21083010083)

#### **Lecture:**

Muhammad Idhom, SP., S.Kom., M.Kom

**DEPARTEMENT OF DATA SCIENCE  
COMPUTER SCIENCE FACULTY  
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”  
JAWA TIMUR  
2022**

## Penjelasan Hasil Dokumentasi Operasi Multiprocessing

Melakukan pemrograman paralel(yang merupakan salah satu konsep dasar sistem operasi) dengan Multiprocessing. Pemrograman paralel adalah sebuah teknik eksekusi perintah yang mana dilakukan secara bersamaan pada CPU. Pemrograman paralel sederhana ini dapat juga diterapkan menggunakan Python.

### Manfaat Multiprocessing:

- Tidak berbagi sumber daya memori
- Menggunakan CPU untuk komputasi
- Tidak memerlukan sinkronisasi memori
- Memerlukan sumber daya memori dan waktu yang tidak sedikit.

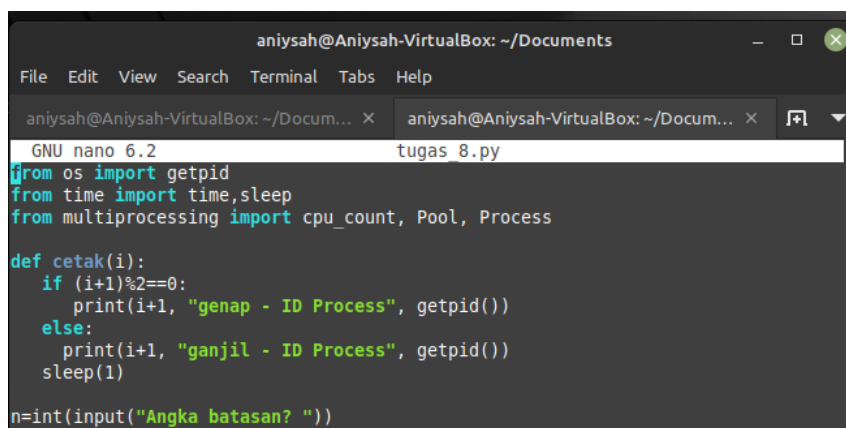
### Penjelasan Soal Latihan :

Buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap dengan menggunakan pemrosesan paralel

### Ketentuan syntax/batasan:

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

1. Langkah awal, create file python dengan code **tugas\_8.py**



```
anisyah@Anisyah-VirtualBox: ~/Documents
File Edit View Search Terminal Tabs Help

anisyah@Anisyah-VirtualBox: ~/Docum... X anisyah@Anisyah-VirtualBox: ~/Docum... X
GNU nano 6.2 tugas_8.py
from os import getpid
from time import time,sleep
from multiprocessing import cpu_count, Pool, Process

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

n=int(input("Angka batasan? "))
```

- Create code sesuai dengan code operasi Multiprocessing
- Untuk awalan import library nya terlebih dahulu.
- Ada **getpid** digunakan untuk mengambil ID proses, **time** mengambil waktu, sleep memberi jeda waktu dan **cpu\_count** melihat jumlah CPU.
- Fungsi sleep digunakan untuk memberi jeda waktu (detik).

- Lalu, untuk **pool** merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada computer.
- Sedangkan **process** merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada computer.

## 2. Inisialisasi fungsi

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

n=int(input("Angka batasan? "))
```

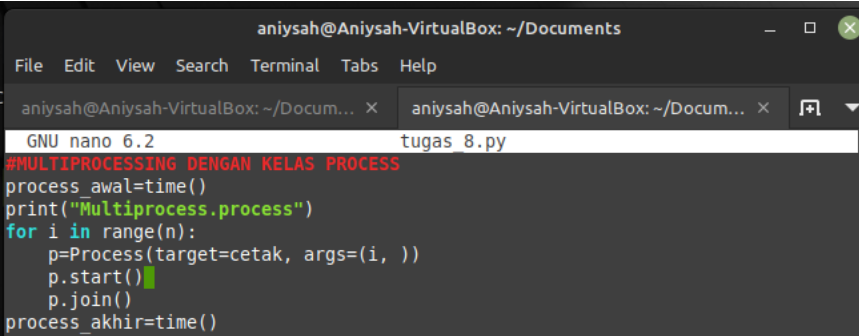
- Kita dapat inisialisasi terlebih dahulu dan fungsi ini digunakan untuk mencetak angka variabel **i** berserta ID proses sejumlah parameter yang diberikan.
- Karena disini ketentuannya menggunakan angka ganjil dan genap jadi ketik code if else dan ada code sleep untuk memberikan jeda tiap detik setiap parameter yang diberikan.
- Lalu, menggunakan notasi **(i+1)%2==0** untuk mengetahui apakah nanti angka yang diinput genap atau ganjil.
- Setelah itu, variabel **n** digunakan untuk input angka dan nilai batasan.

## 3. Pemrosesan sekuensial

```
#SEKUENSIAL
sekuensial_awal = time()
print("Sekuensial")
for i in range(n):
    cetak(i)
sekuensial_akhir=time()
```

- Yang awal ada code waktu sebelum eksekusi lalu, print parameter "Sekuensial"
- Proses nya menggunakan looping **for in range** dan memanggil fungsi cetak yang nanti kita input. Dengan mencetak angka ganjil atau genap dengan id nya masing-masing.
- Selanjutnya sekunsial\_akhir untuk mendapatkan durasi waktu setelah eksekusi.

## 4. Multiprocessing kelas Process



```
aniysah@Aniysah-VirtualBox: ~/Documents
File Edit View Search Terminal Tabs Help
aniysah@Aniysah-VirtualBox: ~/Docum... x aniysah@Aniysah-VirtualBox: ~/Docum... x
GNU nano 6.2 tugas 8.py
#MULTIPROCESSING DENGAN KELAS PROCESS
process_awal=time()
print("Multiprocess.process")
for i in range(n):
    p=Process(target=cetak, args=(i, ))
    p.start()
    p.join()
process_akhir=time()
```

- Pada process ini langkah-langkah nya sama seperti tahap sekuensial
- Selanjutnya akan ada proses dijalankan menggunakan looping for sebanyak angka yang dimasukkan dan menggunakan fungsi cetak yang sudah kita isi di awal.

- Code p.join() digunakan untuk menggabungkan proses agar tidak loncat ke proses sebelumnya. Sehingga menghasilkan id proses yang berbeda
- Process\_akhir untuk mendapatkan durasi waktu proses setelah dijalankan.

## 5. Multiprocessing kelas Pool

```
#MULTIPROCESSING DENGAN KELAS POOL
pool_awal=time()
pool = Pool()
print("Multiprocess.pool")
pool.map(cetak, range(0,n))
pool.close()
pool_akhir=time()
```

- Disini ada variabel pool awal, lalu proses eksekusi dengan pool.map dan range nya dimulai dari 0-n.
- fungsi map() digunakan untuk memetakan pemanggilan fungsi cetak ke dalam setiap CPU yang tersedia sebanyak 0-n kali. Lalu n ini merupakan inputan batasan dari user
- Setelah itu, pool akhir untuk waktu setelah eksekusi.

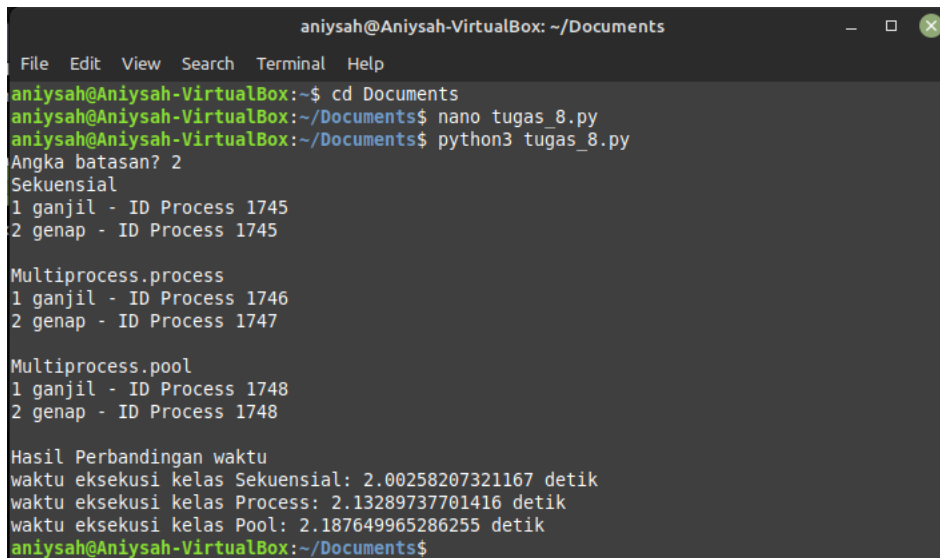
## 6. Pemanggilan seluruh proses

```
#BANDINGKAN WAKTU EKSEKUSI
print("Hasil Perbandingan waktu")
print("waktu eksekusi kelas Sekuensial:", sekuensial_akhir - sekuensial_awal, "detik")
print("waktu eksekusi kelas Process:", process_akhir - process_awal, "detik")
print("waktu eksekusi kelas Pool:", pool_akhir - pool_awal, "detik")
```

^G Help    ^O Write Out    ^W Where Is    ^K Cut    ^T Execute    ^G Location  
 ^X Exit    ^R Read File    ^\ Replace    ^U Paste    ^J Justify    ^\_ Go To Line

- Proses terakhir adalah eksekusi untuk perbandingan seluruh proses dengan code print.
- Ketentuan waktu eksekusi sekuensial dengan memanggil variabel sekuensial\_akhir dikurangi sekuensial\_awal,
- Waktu eksekusi kelas proses dengan memanggil variabel process\_akhir dikurangi process\_awal,
- Lalu yang terakhir, waktu eksekusi kelas pool dengan memanggil variabel pool\_akhir dikurangi variabel pool\_awal

## 7. Hasil Output program Multiprocessing



```
aniysah@Aniysah-VirtualBox: ~/Documents
File Edit View Search Terminal Help
aniysah@Aniysah-VirtualBox:~$ cd Documents
aniysah@Aniysah-VirtualBox:~/Documents$ nano tugas_8.py
aniysah@Aniysah-VirtualBox:~/Documents$ python3 tugas_8.py
Angka batasan? 2
Sekuensial
1 ganjil - ID Process 1745
2 genap - ID Process 1745

Multiprocess.process
1 ganjil - ID Process 1746
2 genap - ID Process 1747

Multiprocess.pool
1 ganjil - ID Process 1748
2 genap - ID Process 1748

Hasil Perbandingan waktu
waktu eksekusi kelas Sekuensial: 2.00258207321167 detik
waktu eksekusi kelas Process: 2.13289737701416 detik
waktu eksekusi kelas Pool: 2.187649965286255 detik
aniysah@Aniysah-VirtualBox:~/Documents$
```

- Setelah itu, kita dapat runninn dengan code **python3 tugas\_8.py**
- Akan muncul inputan untuk batasan angka nya kita bisa isi sesuai keinginan misalnya angka 2.
- Lalu, akan muncul proses **sekuensial** hasil dari print id process yang diinputkan
- Selanjutnya **multiprocessing kelas process** hasil dari print id process, masing-masing bilangan dengan id yang berbeda. Karena tiap running fungsi cetak dilakukan oleh satu process saja.
- Kemudian tahap **multiprocessing kelas pool** hasil dari print id process pada masing-masing bilangan yang tadi kita input.
- Yang terakhir ada output dari hasil perbandingan waktu, dimana akan terlihat berapa detik waktu yang dilakukan saat multiprocessing berjalan. Pada proses eksekusi kelas process itu waktunya lebih lama sedikit dibanding yang lainnya. Karena saat melakukan process, tahap ini memanggil tiap fungsi cetak hanya dengan satu proses saja.