

Nama : Aisyah Kirana Putri Isyanto  
Kelas : Sistem Operasi – A  
NPM : 21083010065

## Tugas 8

### Soal latihan :

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

Dengan batasan sebagai berikut :

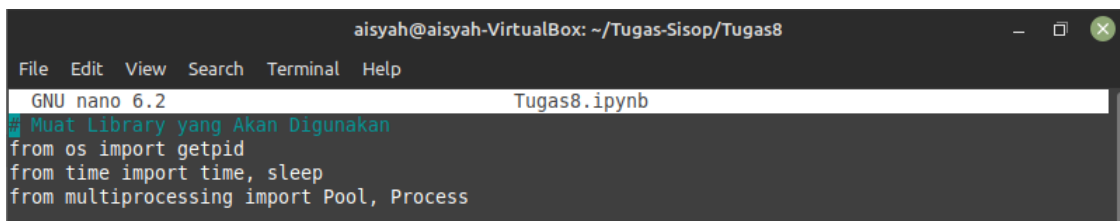
- Nilai yang dijadikan argumen pada fungsi `sleep()` adalah satu detik.
- Masukkan jumlah'nya satu dan berupa bilangan bulat.
- Masukkan adalah batas dari perulangan tersebut.
- Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan paralel.

### Langkah-langkah :

1. Buat file berformat `.ipynb` dengan menggunakan command `nano`

```
aisyah@aisyah-VirtualBox: ~/Tugas-Sisop/Tugas8$ nano Tugas8.ipynb
```

2. Ketikkan script



```
aisyah@aisyah-VirtualBox: ~/Tugas-Sisop/Tugas8
File Edit View Search Terminal Help
GNU nano 6.2 Tugas8.ipynb
Muat Library yang Akan Digunakan
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process
```

### Penjelasan Script :

- `getpid` digunakan untuk mengambil ID proses
- `time` digunakan untuk mengambil waktu(detik)
- `sleep` digunakan untuk memberi jeda waktu(detik)
- `Pool` adalah sebuah class pada library `multiprocessing` yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer.
- `Process` adalah sebuah class pada library `multiprocessing` yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.

```
#Inisialisasi Fungsi
batas = int(input("Masukkan batas angka : "))

def cetak(i):
    for i in range(batas):
        if i % 2 == 0:
            print(f"angka ganjil-{i+1}", " - ID proses", getpid() )
        else:
            print(f"angka genap-{i+1}", " - ID proses", getpid() )
    sleep(1)
```

Penjelasan Script :

- Yang pertama, buat variabel batas yang digunakan sebagai batas akhir program.
- Lalu, buat fungsi cetak dimana nilai variabel i berada dalam batas yang tadi telah dibuat. Pada kondisi saat nilai variabel i dilakukan perhitungan modulus 2 hasilnya 0, maka akan dicetak teks angka ganjil beserta ID prosesnya yang didapatkan dari getpid(). Jika hasilnya selain 0, maka akan dicetak teks angka genap beserta ID prosesnya yang didapat dari getpid().
- Setelah itu, gunakan function sleep untuk memberi jeda selama 1 detik sebelum dilanjutkan ke perintah selanjutnya.

```
# 1 - Pemrosesan Sekuensial
print("Pemrosesan Sekuensial")
sekuensial_awal = time()

for i in range (1):
    cetak(i)

sekuensial_akhir = time()
print(" ")
```

Penjelasan Script :

- Buat variabel sekuensial\_awal untuk mendapatkan waktu sebelum eksekusi.
- Untuk variabel i dalam range 1, maka cetak nilai variabel i
- Setelah itu, buat variabel bernama sekuensial\_akhir untuk mendapatkan waktu setelah eksekusi.

```
# 2- Multiprocessing dengan Kelas Process
print("Multiprocessing dengan Kelas Process")
kumpulan_proses=[]
proses_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()

proses_akhir=time()
print(" ")
```

Penjelasan Script :

- Buat variabel kumpulan\_proses untuk menampung proses-proses yang akan dimuat.

- Kemudian, buat variabel `proses_awal` untuk mendapatkan waktu awal pemrosesan.
- Untuk variabel `i` pada `range`, maka akan dilakukan proses pencetakan dari fungsi cetak. Tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain.
- Kumpulan proses harus ditampung dan digabung menjadi satu (`p.join()`) agar tidak merambah ke proses selanjutnya.
- Buat variabel `proses_akhir` untuk mendapatkan waktu akhir pemrosesan.

```
# 3 - Multiprocessing dengan Kelas Pool
print("Multiprocessing dengan Kelas Pool")
pool_awal = time()

pool = Pool()
pool.map(cetak, range(0,1))
pool.close()

pool_akhir = time()
print(" ")
```

Penjelasan Script :

- Variabel `pool_awal` digunakan untuk mendapatkan waktu awal pemrosesan
- Inisialisasikan metode `pool` dalam variabel `pool`.
- Fungsi `map()` itu memetakan pemanggilan fungsi cetak ke dalam 4 CPU sebanyak 1 kali.
- `pool.close()` menandakan proses telah selesai. Kemudian buat variabel `pool_akhir` agar mendapatkan waktu akhir pemrosesan

```
# Waktu Eksekusi
print("Waktu eksekusi Sekuensial          : ",sekuensial_akhir - sekuensial_awal,"detik")
print("Waktu eksekusi Multiprocessing.Process : ",proses_akhir - proses_awal, "detik")
print("Waktu eksekusi Multiprocessing.Pool    : ",pool_akhir - pool_awal, "detik")
```

Penjelasan Script :

Dilakukan proses untuk mendapatkan lama waktu eksekusi pada setiap pemrosesan dengan cara pengurangan waktu awal dengan waktu akhir tiap proses dan dicetak dalam satuan detik.

### 3. Menjalankan Script

Sebelumnya, user diminta memasukkan batas angka yang akan diolah. Kemudian tekan enter untuk menampilkan hasilnya seperti gambar di bawah.

```
aisyah@aisyah-VirtualBox:~/Tugas-Sisop/Tugas8$ python3 Tugas8.ipynb
Masukkan batas angka : 3
Pemrosesan Sekuensial
angka ganjil-1 - ID proses 2571
angka genap-2 - ID proses 2571
angka ganjil-3 - ID proses 2571

Multiprocessing dengan Kelas Process
angka ganjil-1 - ID proses 2572
angka genap-2 - ID proses 2572
angka ganjil-3 - ID proses 2572

Multiprocessing dengan Kelas Pool
angka ganjil-1 - ID proses 2573
angka genap-2 - ID proses 2573
angka ganjil-3 - ID proses 2573

Waktu eksekusi Sekuensial : 1.0015976428985596 detik
Waktu eksekusi Multiprocessing.Process : 1.0237197875976562 detik
Waktu eksekusi Multiprocessing.Pool : 1.0902671813964844 detik
aisyah@aisyah-VirtualBox:~/Tugas-Sisop/Tugas8$
```