

Nama : Kinanthi Putri Ariyani  
NPM : 21083010047  
Mata Kuliah : Sistem Operasi  
Kelas : B

## Multiprocessing

Pemrograman paralel adalah sebuah teknik eksekusi perintah yang mana dilakukan secara bersamaan pada CPU. *Multiprocessing* terdiri dari 2 jenis yaitu *parallel processing* dan *serial processing*.

Manfaat Multiprocessing antara lain :

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

## Soal Latihan

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.

```
kinanthi@Kinanthi-Linux:~$ nano Tugas_8.py
kinanthi@Kinanthi-Linux:~$ python3 Tugas_8.py
Masukkan bilangan :
3

Sekuensial
1 Ganjil - ID proses 2142
2 Genap - ID proses 2142
3 Ganjil - ID proses 2142

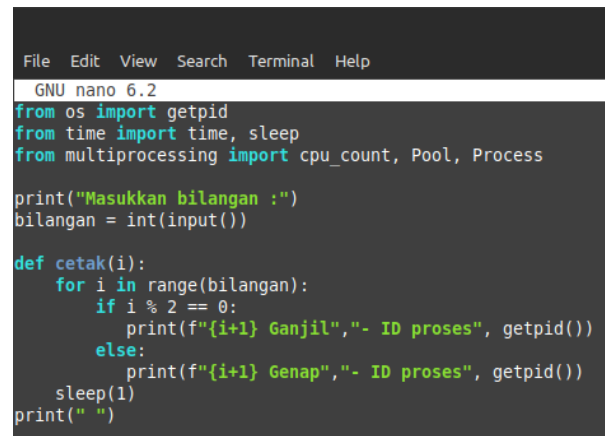
multiprocessing.Process
1 Ganjil - ID proses 2143
2 Genap - ID proses 2143
3 Ganjil - ID proses 2143

multiprocessing.Pool
1 Ganjil - ID proses 2145
2 Genap - ID proses 2145
3 Ganjil - ID proses 2145

waktu eksekusi sekuensial : 1.0020570755004883 detik
waktu eksekusi multiprocessing.Process : 1.0077176094055176 detik
waktu eksekusi multiprocessing.Pool : 1.0207712650299072 detik
kinanthi@Kinanthi-Linux:~$
```

1. Pertama membuat file .py menggunakan perintah **nano namafile.py**, disini saya menggunakan nama file “Tugas\_8.py”
2. Setelah itu kita buat syntax / codingan di dalam file tersebut
3. Kita panggil file “Tugas\_8.py” menggunakan perintah **python3 namafile.py**
4. Setelah itu keluarlah output seperti yang diinginkan. Terdapat jeda 1 detik untuk menuju ke nilai argument yang selanjutnya

Berikut adalah syntax yang saya buat pada file “Tugas\_8.py” :

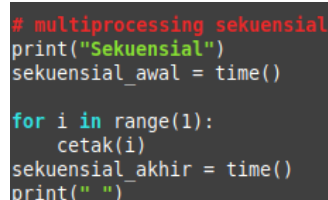


```
File Edit View Search Terminal Help
GNU nano 6.2
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

print("Masukkan bilangan :")
bilangan = int(input())

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

1. Masukkan library yang akan digunakan yaitu getpid untuk mengambil ID proses, time untuk mengambil waktu (detik), sleep untuk memberi jeda waktu (detik), cpu\_count untuk melihat jumlah CPU, pool untuk melakukan pemrosesan paralel menggunakan proses sebanyak jumlah CPU pada komputer, process untuk melakukan pemrosesan paralel menggunakan proses secara beruntun pada komputer
2. Deklarasikan fungsi print dan syntax bilangan agar dapat memasukkan bilangan yang nantinya akan kita masukkan
3. Syntax def cetak adalah untuk mendeklarasikan mana angka yang termasuk bilangan ganjil / genap beserta ID proses sejumlah parameter yang digunakan dan sleep(1) yang artinya jeda 1 detik untuk menuju ke nilai argument selanjutnya



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

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

4. Pada multiprocessing sekuensial menggunakan looping for dan if else disini terdapat syntax sekuensial\_awal = time() untuk mendapatkan waktu sebelum di eksekusi lalu syntax for adalah berlangsungnya proses sekuensial, lalu kita pasang range(1) agar output yang dikeluarkan hanya 1x. Pada syntax sekuensial\_akhir = time() untuk mendapatkan waktu setelah di eksekusi. Lalu kita ketikkan print("") agar terdapat spasi dengan output yang nantinya kita eksekusi.

```
# multiprocessing process
print("multiprocessing.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(" ")
```

5. Pada multiprocessing process awalnya kita tampung proses-prosesnya, setelah itu kita berikan syntax agar mendapatkan waktu sebelum di eksekusi. Lalu proses berlangsung pada syntax for dan menggunakan range(1). Pada syntax for i in kumpulan\_proses: digunakan untuk menggabungkan proses-proses agar tidak loncat ke proses yang sebelumnya.

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

6. Pada multiprocessing pool kita 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")
```

7. Pada syntax di atas adalah kita gunakan syntax ...\_akhir - ...\_awal untuk mendapatkan waktu eksekusinya dan kita bandingkan diantara 3 argumen tersebut (sekuensial, multiprocessing process, dan multiprocessing pool)