Nama: Rizqii Amaliyah M.

NPM : 21083010063

## MULTIPROCESSING

Multiprocessing python adalah pengantar singkat untuk fenomena multiprocessing. Penanganan proses paralel satu sama lain ini dicapai melalui jumlah CPU. Semakin banyak jumlah CPU yang lebih besar akan menjadi peluang multiprocessing yang baik. Untuk pertama membuat file terlebih dahulu dengan command `nano` format `.py`

```
rizqiiamaliyah@rizqiiamaliyah-VirtualBox:~$ nano Tugas_8.py
```

```
  GNU nano 6.2                                  Tugas_8.py *
# Memuat built-in libraries yang akan digunakan
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process

# Input bilangan dan inisialisasi function
a = int(input("Input Bilangan: "))

def cetak(x):
    bil = x % 2
    if bil == 0:
       print(x, "Genap - ID proses", getpid())
    else:
       print(x, "Ganjil - ID proses", getpid())
    sleep(1)

# Sekuensial
print("\nSekuensial")
sekuensial_awal = time()

for x in range(1, a + 1):
    cetak(x)

sekuensial_akhir = time()
```

- Baris 2-4 ialah untuk mengimpor built-in libraries yang akan digunakan dalam program.
- Baris 7-15 ialah penginputan bilangan dan inisialisasi function. Tujuan dari input bilangan ini ialah untuk menginputkan suatu bilangan yang digunakan sebagai batasan. `bil = x % 2` ini melakukan perhitungan modulus 2 terhadap nilai variabel `x` untuk memeriksa bilangan merupakan bilangan genap atau ganjil. `if bil == 0` jika hasil perhitungan modulus adalah 0, maka bilangan `x` merupakan bilangan genap. `sleep(1)` adalah function untuk memberikan jeda selama 1 detik sebelum melanjutkan eksekusi perintah berikutnya.
- Baris 18-24 ialah sekuensial. `\n` digunakan untuk mencetak baris baru. `sekuensial_awal = time()` dan `sekuensial_akhir = time()` untuk mengambil waktu yang disimpan dalam variabel.

```
  GNU nano 6.2                                  Tugas_8.py *
# Multiprocessing Process
print("\nMultiprocessing.Process")
kumpulan_process = []
process_awal = time()

for x in range(1, a + 1):
    p = Process(target=cetak, args=(x,))
    kumpulan_process.append(p)
    p.start()

for x in kumpulan_process:
    p.join()

process_akhir = time()

# Multiprocessing Pool
print("\nMultiprocessing.Pool")
pool_awal = time()

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

pool_akhir = time()
```

- Baris 27-39 ialah multiprocessing process. `\n` digunakan untuk mencetak baris baru. `process_awal = time()` dan `process_akhir = time()` untuk mengambil waktu yang disimpan dalam variabel. Menginisialkan list kosong `[]` dengan nama variabel `kumpulan_process`.

- Baris 42-49 ialah multiprocessing pool. `\n` digunakan untuk mencetak baris baru. `pool_awal = time()` dan `pool_akhir = time()` untuk mengambil waktu yang disimpan dalam variabel. Menginisialkan kelas `Pool ()` dengan variabel `pool`.

```
# 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")
```

- Baris 52-54 ialah waktu eksekusi. Mencetak nilai durasi wajtu untuk setiap proses melalui perhitungan waktu akhir setiap proses dikurangi dengan waktu awal setiap proses dengan menggunakan hasil waktu dalam detik.

```
rizqiiamaliyah@rizqiiamaliyah-VirtualBox:~$ python3 Tugas_8.py
Input Bilangan: 3

Sekuensial
1 Ganjil - ID proses 2800
2 Genap - ID proses 2800
3 Ganjil - ID proses 2800

Multiprocessing.Process
2 Genap - ID proses 2802
1 Ganjil - ID proses 2801
3 Ganjil - ID proses 2803

Multiprocessing.Pool
1 Ganjil - ID proses 2804
2 Genap - ID proses 2804
3 Ganjil - ID proses 2804
Waktu eksekusi sekuensial               :  3.0027859210968018 detik
Waktu eksekusi multiprocessing.process  :  1.0212271213531494 detik
Waktu eksekusi multiprocessing.pool     :  3.0539982318878174 detik
rizqiiamaliyah@rizqiiamaliyah-VirtualBox:~$
```

Gambar diatas ialah menunjukkan hasil atau output dari program yang sudah dijalankan. Untuk pertama menginputkan bilangan 3 sebagai batasan. Dapat disimpulkan bahwa

proses sekuensial leboh lambat dibanding proses multiprocessing, tetapi kita tidak perlu menggunakan proses multiprocessing terus menerus dapat menggunakan metode sesuai kapasitas.