Nama : Ellexia Leonie G.

NPM : 21083010027

Kelas : Sistem Operasi-A

Multiprocessing

1. Membuat file python “Tugas_8.py”

```
ellexia@ellexia-VirtualBox:~/Documents/Tugas-Sisop/Tugas-8$ nano Tugas_8.py
```

2. Meng-*import libraries* yang akan digunakan

```
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

- **getpid** digunakan untuk mengambil ID proses
- **time** digunakan untuk mengambil waktu(detik)
- **sleep** digunakan untuk memberi jeda waktu(detik)
- **cpu_count** digunakan untuk melihat jumlah CPU
- **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

3. Menginisialisasikan fungsi “Genap” dan “Ganjil”

```
#Inisialisasi fungsi
def cetak(i):
    if (i+1)%2==0:
      print(i+1, "Genap - ID proses", getpid())
    else:
      print(i+1,"Ganjil - ID proses", getpid())
    sleep(1)
```

Jika bilangan + 1 kemudian habis dibagi 2, maka bilangan tersebut “Genap”.
Jika tidak, maka bilangan tersebut adalah “Ganjil”.

4. Membuat inputan/ masukkan bilangan

```
a=int(input("Masukkan batas perulangan: "))
```

5. Membuat pemrograman **sekuensial**

```
#Pemrosesan sekuensial
print("Sekuensial")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
sekuensial_awal = time()
# PROSES BERLANGSUNG
for i in range(a):
    cetak(i)
# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()
```

- sekuensial_awal digunakan untuk mendapatkan waktu sebelum eksekusi.

- Membuat looping sebanyak nilai yang diinputkan oleh user (a). Cetak(i) digunakan untuk mencetak setiap angka ganjil dan genap (yang sudah diinisialisasikan sebelumnya/ pada proses inisialisasi).
- sekuensial_akhir digunakan untuk mendapatkan waktu setelah eksekusi.

6. Membuat pemrograman **multiprocessing.Process**

```
#Multiprocessing dengan kelas Process
print("multiprocessing.Process")
# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()
# PROSES BERLANGSUNG
for i in range(a):
    p = Process(target=cetak, args=(i,))
    kumpulan_proses.append(p)
    p.start()
# UNTUK MENGGABUNGKAN PROSES-PROSES AGAR TIDAK LONCAT KE PROSES SEBELUM'NYA
for i in kumpulan_proses:
    p.join()
# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
process_akhir = time()
```

Kumpulan proses ditampung dan digabung menjadi satu agar tidak merambah ke proses selanjutnya.

7. Membuat pemrograman multiprocessing.Pool

```
#Multiprocessing dengan kelas Pool
print("multiprocessing.Pool")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()
# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(0,a))
pool.close()
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()
```

8. Membandingkan waktu eksekusi dari setiap proses pemrograman

```
#Bandingkan 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")
```

**Output**

```
ellexia@ellexia-VirtualBox:~/Documents/Tugas-Sisop/Tugas-8$ python3 Tugas_8.py
Masukkan batas perulangan: 5
Sekuensial
1 Ganjil - ID proses 3561
2 Genap - ID proses 3561
3 Ganjil - ID proses 3561
4 Genap - ID proses 3561
5 Ganjil - ID proses 3561
multiprocessing.Process
1 Ganjil - ID proses 3563
3 Ganjil - ID proses 3565
2 Genap - ID proses 3564
4 Genap - ID proses 3566
5 Ganjil - ID proses 3567
multiprocessing.Pool
1 Ganjil - ID proses 3568
2 Genap - ID proses 3568
3 Ganjil - ID proses 3568
4 Genap - ID proses 3568
5 Ganjil - ID proses 3568
Waktu eksekusi sekuensial : 5.003666162490845 detik
Waktu eksekusi multiprocessing.Process : 1.021993637084961 detik
Waktu eksekusi multiprocessing.Pool : 5.030869245529175 detik
```

Pada praktik ini, saya memasukkan nilai batasan sebesar 5.

- ID proses pada sekuensial terlihat sama semua. Ini dikarenakan sekuensial akan mengeksekusi pada proses yang sama.
- ID proses pada multiprocessing.Process terlihat berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain.
- Sedangkan ID proses pada multiprocessing.Pool terlihat sama semua, karena pada laptop saya hanya ada 2 CPU.

Dari hasil waktu eksekusi, didapatkan bahwa eksekusi multiprocessing.Process adalah yang tercepat.