

Nama : Fernando Hadi W

NPM : 21083010080

Kelas : Sistem Operasi B

Manfaat Multiprocessing:

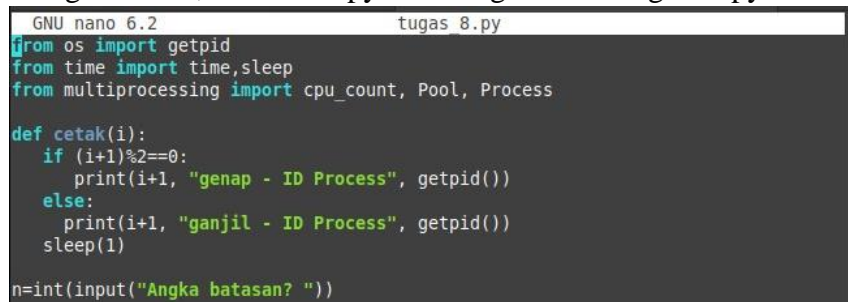
Tidak berbagi sumber daya memori, Menggunakan CPU untuk komputasi, Tidak memerlukan sinkronisasi memori, Memerlukan sumber daya memori dan waktu yang tidak sedikit.

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 pyhton dengan code Tugas_8.py



```
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? "))
```

- a) Create code sesuai dengan code operasi Multiprocessing
- b) Untuk awalan import library nya terlebih dahulu.
- c) Ada getpid digunakan untuk mengambil ID proses, time memngambil waktu, sleep memberi jeda waktu dan cpu_count melihat jumlah CPU.
- d) Fungsi sleep digunakan untuk memberi jeda waktu (detik).
- e) Lalu, untuk pool merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada computer.
- f) Sedangkan process merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada computer.

2. 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 masingmasing.
- Selanjutnya sekunsial_akhir untuk mendapatkan durasi waktu setelah eksekusi.

3. Multiprocessing kelas Process

```
#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.

4. 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.

5. 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")
```

- 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

6. Hasil Output eksekusi file pyhton3

```
fernando@fernando-VirtualBox:~/Pertemuan8$ python3 Tugas_8.py
Angka batasan? 3
Sekuensial
1 ganjil - ID Process 2676
2 genap - ID Process 2676
3 ganjil - ID Process 2676
Multiprocess.process
1 ganjil - ID Process 2678
2 genap - ID Process 2679
3 ganjil - ID Process 2680
Multiprocess.pool
1 ganjil - ID Process 2681
2 genap - ID Process 2682
3 ganjil - ID Process 2683
Perbandingan waktu
Sekuensial: 3.0066356658935547 detik
Kelas Process: 3.019791841506958 detik
Kelas Pool: 1.0202205181121826 detik
```

- Setelah itu, kita dapat runnin dengan code python3 Tugas_8.py
- Akan muncul inputan untuk batasan angka nya kita bisa isi sesuai keinginan misalnya angka 3
- 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.