

Nama : Naufal Baihaqi M

Npm : 21083010077

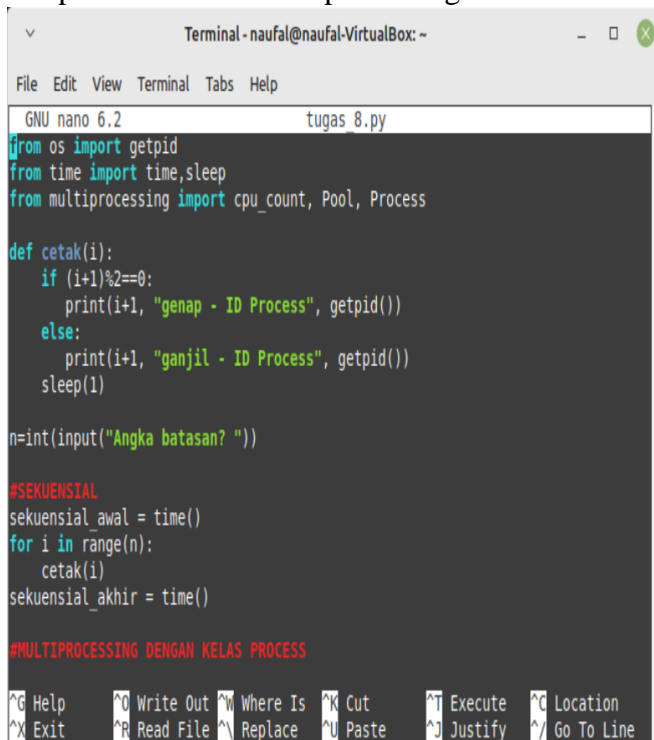
## Multiprocessing

Objektifitas pada modul 7 adalah melakukan pemrograman paralel(yang merupakan salah satu konsep dasar system operasi) dengan Multiprocessing. Pemrograman paralel adalah sebuah teknik eksekusi perintah yang mana dilakukan secara bersamaan pada CPU. Seluruh bahasa pemrograman yang populer dapat melakukan pemrograman paralel dengan modul bawaan atau memang pengaturan defaultnya seperti itu. Praktikan akan melakukan penerapan pemrograman paralel sederhana dengan Python. Kami memilih Python karena bahasa pemrograman tersebut sudah otomatis terinstal di hampir seluruh sistem operasi berbasis Linux selain itu secara default komputasi di Python dilakukan secara sekuensial. Walaupun sekuensial kita dapat menjadikannya paralel dengan fungsi bawaan yang telah disediakan oleh Python

Manfaat Multiprocessing :

1. Menggunakan CPU untuk komputasi
2. Tidak berbagi sumber daya memori
3. Memerlukan sumber daya memori dan waktu yang tidak sedikit
4. Tidak memerlukan sinkronisasi memori input

- Script soal Latihan multiprocessing



```
Terminal - naufal@naufal-VirtualBox: ~
File Edit View Terminal Tabs Help
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? "))

#SEKUENSIAL
sekuensial_awal = time()
for i in range(n):
    cetak(i)
sekuensial_akhir = time()

#MULTIPROCESSING DENGAN KELAS PROCESS

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^I Replace   ^U Paste     ^_ Justify   ^_ Go To Line
```

```

#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()

#MULTIPROCESSING DENGAN KELAS POOL
pool_awal = time()
pool = Pool()
print("Multiprocess.pool")
pool.map(cetak, range(0,n))
pool.close()
pool_akhir = time()

#BANDINGKAN WAKTU EKSEKUSI
print("Perbandingan waktu")
print("Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")

```

- Pengertian script

1. Kita import dulu library yang diperlukan

```

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

```

2. Fungsi ini digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Kita panggil fungsi sleep untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan.

```

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? "))

```

3. Pemrosesan sekuensial

```

#SEKUENSIAL
sekuensial_awal = time()
for i in range(n):
    cetak(i)
sekuensial_akhir = time()

```

#### 4. Multiprocessing dengan 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()
```

- process\_awal adalah variabel untuk mendapatkan waktu awal mulainya proses dijalankan
- process\_akhir adalah variabel untuk mendapatkan waktu berakhirnya proses dijalankan
- Dapat diperhatikan dengan seksama bahwa ID proses tiap memanggil fungsi cetak adalah berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain.
- Lakukan Looping sebanyak angka yang dimasukkan oleh user, dan gunakan fungsi cetak yang sudah kita isi di awal untuk mencetak setiap angka ganjil atau genap dengan id proses masing – masing
- p.start() digunakan untuk mengeksekusi fungsi cetak di kelas process
- p.join() digunakan agar proses ditunggu hingga proses sebelumnya selesai. Sehingga akan menghasilkan id proses yang berbeda – beda tiap prosesnya
- Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

#### 5. Multiprocess dengan 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()
```

- pool\_awal adalah variabel untuk mendapatkan waktu awal mulainya proses dijalankan
- pool\_akhir adalah variabel untuk mendapatkan waktu berakhirnya proses proses dijalankan

- fungsi map() digunakan untuk memetakan pemanggilan fungsi cetak ke dalam setiap CPU yang tersedia sebanyak 0-n kali yang mana 'n' adalah inputan batasan dari user.
- Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

6. Bandingkan waktu eksekusi

```
#BANDINGKAN WAKTU EKSEKUSI
print("Perbandingan waktu")
print("Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")
```

Kemudian print waktu eksekusi sekuensial, kelas process, dan kelas pool untuk melihat hasil waktu berapa detiknya. Dan akan menghasilkan output seperti dibawah ini.

```
naufal@naufal-VirtualBox:~$ python3 tugas_8.py
Angka batasan? 3
1 ganjil - ID Process 2125
2 genap - ID Process 2125
3 ganjil - ID Process 2125
Multiprocess.process
1 ganjil - ID Process 2126
2 genap - ID Process 2127
3 ganjil - ID Process 2128
Multiprocess.pool
1 ganjil - ID Process 2129
2 genap - ID Process 2129
3 ganjil - ID Process 2129
Perbandingan waktu
Sekuensial : 3.003021478652954 detik
Kelas Process : 3.0176587104797363 detik
Kelas Pool : 3.028733730316162 detik
naufal@naufal-VirtualBox:~$
```