Nama : Alya Setya Paramita

NPM : 21083010046

Kelas : Sistem Operasi A

Soal Latihan

Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap!

- Nilai yang dijadikan argumen 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.

Jawaban :

Pada latihan soal ini akan menentukan suatu bilangan termasuk dalam bilangan ganjil atau bilangan bulat. Berikut langkah-langah membuat programnya dan langkahnya sudah terdapat pada keterangan setiap gambar.

Membuat file **nano Tugas_8.py**

```
mint@mint:~/Documents$ nano Tugas_8.py
```

Mengetikkan script seperti di bawah ini sesuai langkah-langkahnya

➢

```
# IMPORT BUILT-IN LIBRARY
# getpid -> mengambil ID proses
# time -> mengambil waktu
# sleep -> memberi jeda waktu
# cpu_count -> jumlah cpi
# pool -> cpu pada komputer
# process -> pemrosesan paralel
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

➢

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

➢

```
# INPUT NILAI
n = int(input("Masukkan Batas Nilai : "))
```

➢

```
# SEKUENSIAL
print("Sekuensial")
# Mendapatkan waktu sebelum eksekusi
sekuensial_awal = time()
# Proses berlangsung
for i in range(n):
    cetak(i)
# Mendapatkan waktu setelah eksekusi
sekuensial_akhir = time()
```

➢

```
# MULTIPROCESSING -> PROCESS
print("Multiprocessing.Process")
# Menampung proses
kumpulan_proses = []
# Mendapatkan waktu sebelum eksekusi
proses_awal = time()
# Proses berlangsung
for i in range(n):
    p = Process(target = cetak, args = (i, ))
    kumpulan_proses.append(p)
    p.start()
# Menggabungkan proses agar tidak loncat ke proses sebelumnya
for i in kumpulan_proses:
    p.join()
# Mendapatkan waktu setelah eksekusi
proses_akhir = time()
```

➢

```
# MULTIPROCESSING -> POOL
print("Multiprocessing.Pool")
# Mendapatkan waktu sebelum eksekusi
pool_awal = time()
# Proses berlangsung
pool = Pool()
pool.map(cetak, range(0,n))
pool.close
# Mendapatkan waktu setelah eksekusi
pool_akhir = time()
```

➢

```
# PERBANDINGAN WAKTU EKSEKUSI
print("Waktu eksekusi Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi Multiprocessing.Process:", proses_akhir - proses_awal, "detik")
print("Waktu eksekusi Multiprocessing.Pool:", pool_akhir - pool_awal, "detik")
```

Jalankan dengan perintah **python3 Tugas_8.py** sehingga menghasilkan output seperti ini. Dengan keterangan dalam terdapat perintah memasukkan batasan nilai, maka saya memasukkan nilai 3 sehingga akan muncul perulangan sebanyak 3 kali. Dan didapatkan bahwa sekuensial merupakan yang paling lambat dibandingkan dengan multiprocessing proses maupun pool.

```
mint@mint:~/Documents$ python3 Tugas_8.py
Masukkan Batas Nilai : 3
Sekuensial
1 Ganjil - ID proses 2144
2 Genap - ID proses 2144
3 Ganjil - ID proses 2144
Multiprocessing.Process
1 Ganjil - ID proses 2145
2 Genap - ID proses 2146
3 Ganjil - ID proses 2147
Multiprocessing.Pool
1 Ganjil - ID proses 2148
2 Genap - ID proses 2148
3 Ganjil - ID proses 2148
Waktu eksekusi Sekuensial : 3.0048742294311523 detik
Waktu eksekusi Multiprocessing.Process: 1.0300781726837158 detik
Waktu eksekusi Multiprocessing.Pool: 3.086273193359375 detik
```