

Nama : Fathima Zahrah

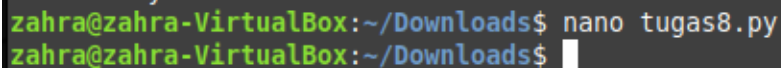
NPM : 21083010043

Mata Kuliah / Kelas : Sistem Operasi / A

Laporan Tugas Pertemuan 8

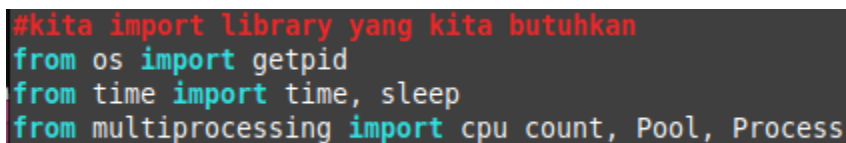
Tugas : Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap!

Pada tugas kali kami diminta membuat program yang dapat menentukan sebuah bilangan itu ganjil atau genap menggunakan pemrosesan paralel dengan Bahasa python di linux. Untuk membuat hal tersebut yang harus kita siapkan adalah terminal linux kita. Setelah terbuka maka kita akan membuat file python terlebih dahulu pada direktori yang kita inginkan menggunakan fungsi nano kemudian nama_file.py.



```
zahra@zahra-VirtualBox:~/Downloads$ nano tugas8.py
zahra@zahra-VirtualBox:~/Downloads$
```

Pada file yang saya buat, file tersebut bernama tugas8.py. Setelah selesai dan kita enter maka akan muncul text editor dari linux yang tandanya sudah siap untuk kita tulisi kode. Untuk membuat pemrograman tugas diatas, yang kita harus lakukan terlebih dahulu adalah me-import library - library yang kita perlukan.



```
#kita import library yang kita butuhkan
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

Gambar diatas merupakan library library yang harus kita import agar kode kita bisa berjalan dengan baik. Pada library os kita mengimport fungsi getpid. Fungsi ini bertujuan untuk mengambil ID Proses saat menjalankan kode kita. Pada library time kita akan mengimport fungsi time dan sleep. Pada fungsi time bertujuan untuk mengambil waktu saat pemrosesan terjadi sedangkan pada fungsi slepp bertujuan untuk memberi jeda waktu. Pada library multiprocessing kita mengimport fungsi cpu_count, pool serta process. Pada fungsi cpu_count bertujuan untuk melihat jumlah cpu yang sedang berfungsi, pada fungsi Pool bertujuan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer, sedangkan pada fungsi process bertujuan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.

Setelah kita mengimport library library diatas, maka selanjutnya kita akan membuat function yang bisa menentukan angka tersebut ganjil atau genap. Namun sebelum itu kita harus membuat kode yang bertujuan untuk mengambil inputan angka yang user inginkan.

```
#untuk mengambil input dari user
i=input("masukkan angka")
i = int(i) #kita rubah jadi integer
```

Untuk mengambil inputan user kita bisa menggunakan fungsi input. Pada kode diatas inputan user ditaruh pada variabel i yang kemudian variabel tersebut dirubah menjadi objek integer. Setelah itu kita membuat function yang menentukan angka ganjil genap.

```
#kita buat function untuk mengenentukan ganjil genap
def gg(i):
    i+=1
    if (i%2) == 0:
        print(i, 'genap - ID Proses', getpid())
    else:
        print(i, 'ganjil - ID Proses', getpid())
    sleep(1)
```

Pada function yang saya buat, saya menamai dengan nama gg (ganjil genap) dengan parameter I, kemudian pada kode selanjutnya i+=1 hal ini bertujuan agar index awal yang dimulai pada saat proses dimulai adalah angka 1 bukan 0 , saat index ke 2 bukan 1 melainkan 2. Hal ini dikarenakan saat kode looping dijalankan looping akan selalu memproses index awal dengan angka 0 bukan 1. Setelah itu pada kode tersebut tertulis apabila angka i modulus 2 menghasilkan angka 0 maka ia termasuk angka genap dan kode akan nmengeluarkan output angka i lalu genap dan disertai ID proses proses terdebut. Namun apabila i modulus 2 bukan menghasilkan angka 0 maka ia termasuk angka ganjil dan kode akan mengeluarkan output angka i, ganjil dan disertai ID proses proses tersebut. Pada baris kode selanjutnya yaitu sleep(1) yang artinya tiap proses tersebut selesai maka saat menuju proses selanjtnya akan terjeda 1 detik.

Setelah function yang kita buat selesai maka kitaa lanjut menulis kode selanjutnya yaitu kode proses sekuensial. Sekuensial merupakan sebuah metode proses dimana proses tersebut dilakukan secara berurutan dan satu persatu dalam pemrosesanya. Berikut kode pemrosesan sekuensial yang telah saya buat:

```
#proses Sekuensial
print('sekuensial')
sekuensial_awal = time()
(
for a in range(i):
    gg(a)

sekuensial_akhir = time()
```

Pada kode diatas yang saya lakukan pertama adalah me-print teks sekuensial agar saat kode ini berjalan kita tahu bahwa output dibawahnya merupakan hasil dari proses sekuensial. Setelah itu terdapat kode yang menggunakan fungsi time seperti gambar diatas, hal ini bertujuan untuk mengambil waktu awal pemrosesan sekuensial terjadi dan menyimpannya pada variabel sekuensial_awal. Pada baris kode selanjutnya merupakan pemrosesan sekuensial inti dilakukan yang mana terdiri dari looping for dan pemanggilan function yang telah kita buat. pada looping ,

index angka akan didefinisikan variabel a dengan range hasil inputan yang user masukkan. setelah index ke akan dilakukan pengecekan ganjil genap melalui function yang telah kita buat tadi. Looping akan terus berjalan sampai index a telah sampai range angka hasil inputan user tadi. setelah looping berakhir kita akan menyimpan waktu tersebut menggunakan fungsi time yang disimpan pada variabel sekuensial_akhir.

Selanjutnya kita akan membuat proses Multi Processing pada kelas process. Multi processing sendiri merupakan proses dimana kita akan mengeksekusi perintah yang dilakukan secara bersamaan atau serentak. Berikut kode multi processing dengan kelas process yang telah saya buat:

```
#proses Multi processing
print('Multi Processing proses')
kumpulan_proses = []

process_awal = time()

for a in range(i):
    p = Process(target=gg, args=(a,))
    kumpulan_proses.append(p)
    p.start()

for a in kumpulan_proses:
    p.join()

process_akhir = time()
```

Pada kode diatas yang saya lakukan pertama adalah me-print teks multi processing proses agar saat kode ini berjalan kita tahu bahwa output dibawahnya merupakan hasil dari proses multi processing kelas proses. Kemudian kita membuat sebuah variabel yang bernama kumpulan proses yang berisi array kosong, hal ini bertujuan untuk menampung hasil dari pemrosesan multi processing. Pada baris selanjutnya kita akan membuat membuat looping for. Looping ini index angka akan didefinisikan dengan variabel a dengan range i yang merupakan hasil inputan user. Kemudian kita definisikan variabel p, dimana variabel ini berisi fungsi process untuk memproses function yang telah kita buat dengan range i. Lalu pada baris selanjutnya memiliki arti bahwa hasil proses tersebut akan ditambahkan di dalam variabel kumpulan_proses. Kemudian untuk menggabungkan proses proses agar tidak loncat ke proses selanjutnya akan dilakukan looping for, yang mana index pada isi array didefinisikan dengan a kemudian variabel p yang kita buat tadi dikumpulkan dengan fungsi join. Setelah proses selesai kita akan mengambil waktu selesai tersebut dan disimpan pada variabel proses akhir.

Lanjut ke proses selanjutnya kita akan membuat multiprocessing dengan kelas pool. Berikut kode yang telah saya buat:

```
#proses multi processing kelas pool
print('multi processing pool')

pool_awal = time()

pool = Pool()
pool.map(gg, range(0,i))
pool.close()

pool_akhir = time()
```

Pertama supaya kita mengetahui bahwa output yang akan dikeluarkan dibawah adalah multiprocessing kelas pool maka kita harus me-print teks terlebih dahulu seperti gambar diatas. Setelah itu kita mengambil waktu awal proses itu terjadi menggunakan fungsi time yang disimpan pada variabel pool_awal. Pada baris selanjutnya kita mendefinisikan fungsi pool terlebih dahulu pada variabel pool hal ini bertujuan agar kita mudah menggunakannya. Fungsi pool ini sendiri memiliki fungsi untuk menjalankan multiprocessing kelas pool. Setelah itu kita panggil fungsi tersebut yang ada pada variabel pool kemudian disertai fungsi map yang berfungsi memetakan pemanggilan fungsi dengan parameter function yang telah kita buat dan variabel hasil inputan user. Lalu setelah pool selesai maka akan disclose menggunakan fungsi close. Setelah itu kita mengambil waktu saat pemrosesan berakhir dengan fungsi time yang disimpan pada variabel pool akhir.

Untuk membandingkan waktu yang dibutuhkan dalam menyelesaikan proses pada 3 sistem processing ini kita dapat membuat program seperti yang telah saya buat dibawah ini:

```
print("Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")
```

Untuk mengeluarkan output tersebut kita akan memakai fungsi print kemudian mengurangi variabel waktu akhir pada tiap proses dengan variabel waktu awal proses pada tiap proses. Yang mana sebelumnya kita telah membuat variabel tersebut diawal sebelum proses pada tiap proses itu terjadi dan sesudahnya.

Setelah kode selesai ditulis maka waktunya bagi kita untuk me-run kode tersebut pada terminal linux kita dengan menuliskan python3 lalu nama_file.

```
zahra@zahra-VirtualBox:~/Downloads$ python3 tugas8.py
```

Setelah kita enter maka kita akan diminta memasukkan bilangan seperti gambar dibawah ini

```
zahra@zahra-VirtualBox:~/Downloads$ python3 tugas8.py
masukkan angka
```

Setelah kita memasukkan angka tersebut. Maka sistem akan memproses ketiga processing tersebut dan mengeluarkan output seperti dibawah ini

```
zahra@zahra-VirtualBox:~/Downloads$ python3 tugas8.py
masukkan angka3
sekuensial
1 ganjil - ID Proses 3410
2 genap - ID Proses 3410
3 ganjil - ID Proses 3410
Multi Processing proses
1 ganjil - ID Proses 3411
3 ganjil - ID Proses 3413
2 genap - ID Proses 3412
multi processing pool
1 ganjil - ID Proses 3414
2 genap - ID Proses 3414
3 ganjil - ID Proses 3414
Sekuensial : 3.004128932952881 detik
Kelas Process : 1.0547401905059814 detik
Kelas Pool : 3.094937801361084 detik
```

Pada Output tersebut menunjukkan bahwa pada sekuensial hanya menjalankan satu proses secara berurutan dikarenakan memiliki ID proses yang sama. Pada multi processing kelas proses memiliki id proses yang berbeda beda hal ini menunjukkan bahwa tiap pemanggilan function ditangani oleh satu proses saja dan selanjutnya ditangani oleh proses yang lain. Sedangkan pada multiprocessing kelas pool mengeluarkan satu jenis id proses dikarenakan dilaptop saya hanya memiliki 1 cpu saja. Untuk waktu yang dihasilkan seharusnya yang memakan waktu lebih banyak yaitu sekuensial akan tetapi dikarenakan pada processing kelas pool hanya dijalankan dengan satu cpu saja maka sedikit lebih lama dan proses tercepat dijalankan oleh multiprocessing kelas process.