

Nama : Rangga Laksana Aryananda

NPM : 21083010036

Kelas : Sistem Operasi B

Prodi : Sains Data

Brief Tugas 8

1. Memahami materi yang ada pada Modul 8 Sistem Operasi_Multiprocessing.pdf
2. Mengerjakan latihan soal, sesuai dengan output yang diinginkan oleh soal. (Tugas_8.py)
3. Menjelaskan program yang telah dibuat kedalam bentuk pdf (Laporan_Tugas_8.pdf)
4. Lalu di upload ke github (tdk perlu push, add file saja)
5. File laporan dokumentasi (.pdf) juga di upload ke github

Soal latihan:

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

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.

Output :

Contoh input :

3

Contoh Output :

```
Sekuensial
1 Ganjil - ID proses ****
2 Genap - ID proses ****
3 Ganjil - ID proses ****

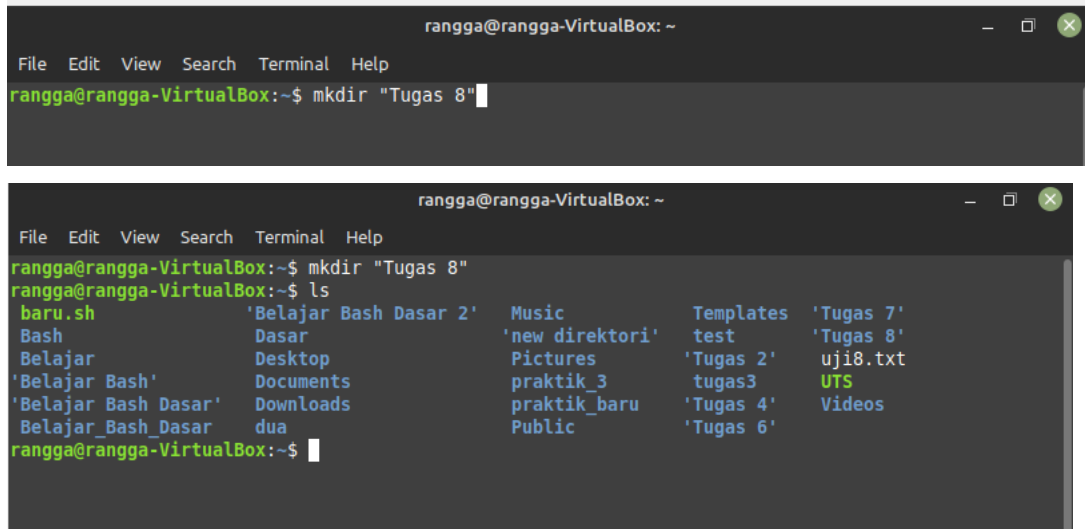
multiprocessing.Process
1 Ganjil - ID proses ****
2 Genap - ID proses ****
3 Ganjil - ID proses ****

multiprocessing.Pool
1 Ganjil - ID proses ****
2 Genap - ID proses ****
3 Ganjil - ID proses ****

Waktu eksekusi sekuensial : ** detik
Waktu eksekusi multiprocessing.Process : ** detik
Waktu eksekusi multiprocessing.Pool : ** detik
```

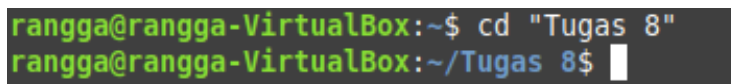
Penyelesaian :

1. Buat direktori untuk menyimpan file tugas 8



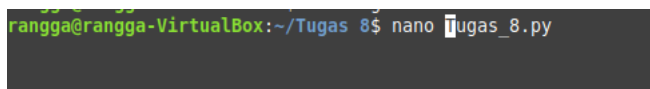
```
rangga@rangga-VirtualBox: ~  
File Edit View Search Terminal Help  
rangga@rangga-VirtualBox:~$ mkdir "Tugas 8"  
  
rangga@rangga-VirtualBox:~$ ls  
baru.sh      'Belajar Bash Dasar 2'  Music      Templates  'Tugas 7'  
Bash         Dasar                   'new direktori' test       'Tugas 8'  
Belajar      Desktop                Pictures    'Tugas 2'  uji8.txt  
'Belajar Bash' Documents              praktik_3   tugas3     UTS  
'Belajar Bash Dasar' Downloads            praktik_baru 'Tugas 4'  Videos  
Belajar_Bash_Dasar dua      Public        'Tugas 6'
```

2. Masuk ke direktori untuk membuat file tugas 8



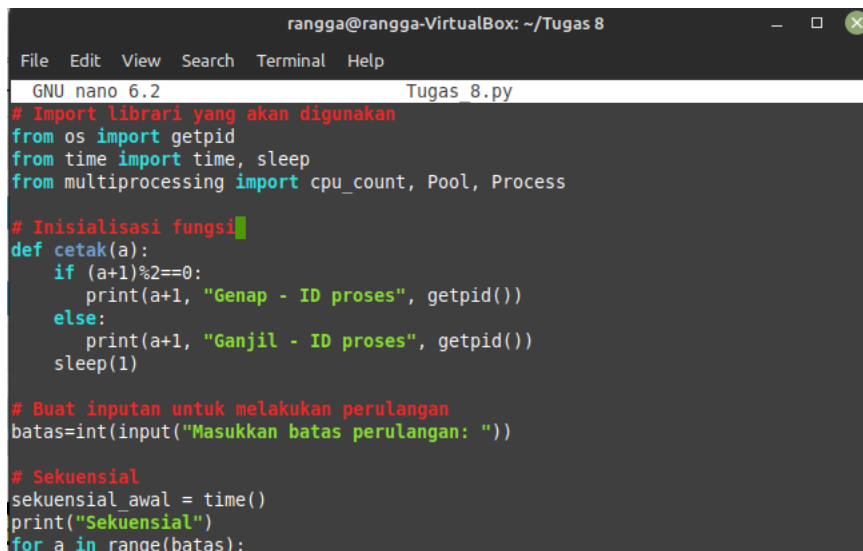
```
rangga@rangga-VirtualBox:~$ cd "Tugas 8"  
rangga@rangga-VirtualBox:~/Tugas 8$
```

3. Buat file baru Tugas_8.py untuk menyimpan script Tugas 8



```
rangga@rangga-VirtualBox:~/Tugas 8$ nano Tugas_8.py
```

4. Masukkan Script pada nano text editor



```
rangga@rangga-VirtualBox: ~/Tugas 8  
File Edit View Search Terminal Help  
GNU nano 6.2 Tugas_8.py  
# Import librari yang akan digunakan  
from os import getpid  
from time import time, sleep  
from multiprocessing import cpu_count, Pool, Process  
  
# Inisialisasi fungsi  
def cetak(a):  
    if (a+1)%2==0:  
        print(a+1, "Genap - ID proses", getpid())  
    else:  
        print(a+1, "Ganjil - ID proses", getpid())  
    sleep(1)  
  
# Buat inputan untuk melakukan perulangan  
batas=int(input("Masukkan batas perulangan: "))  
  
# Sekuensial  
sekuensial_awal = time()  
print("Sekuensial")  
for a in range(batas):
```

Pada percobaan Tugas 8 ini saya membuat suatu script kode python yang akan digunakan untuk mencetak perintah bilangan angka genap dan ganjil sesuai dengan inputan perulangan yang dimasukkan. Angka akan bertambah sesuai dengan perulangan yang dimasukkan sebagai batas

atasnya. Pada Latihan Tugas 8 ini terdapat beberapa library yang saya import dan gunakan. Library tersebut diantaranya:

- Getpid adalah fungsi dari library yang digunakan untuk mendapatkan ID proses dari proses yang memanggil library tersebut.
- Time adalah fungsi dari library yang akan digunakan dalam kode ini, kegunaan dari time sendiri adalah time digunakan untuk mengetahui dan memcetak waktu yang diperlukan suatu proses nantinya.
- Sleep adalah fungsi dari Library yang digunakan untuk menghentikan atau menjeda waktu proses.
- Cpu Count adalah fungsi dari Library yang digunakan untuk mengetahui jumlah dari cpu yang ada.
- 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

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

Setelah melakukan pengimportan library untuk keperluan koding dalam Tugas 8 ini, langkah berikutnya yang dilakukan adalah inisialisasi fungsi. Dalam script kode ini inisialisasi fungsi dilakukan untuk membuat proses pada kode agar terjadi proses untuk mendeteksi dan mengetahui apakah bilangan atau angka ini termasuk bilangan genap atau ganjil. Inisialisasi atau pendaklarasian fungsi diawali dengan def dan nama fungsi (cetak) sesuai dengan aturan python. Kemudian buat kondisi satu dengan if, pada kondisi if ini dilakukan suatu pembacaan dan pendeteksian suatu bilangan genap, jika a (angka) ditambahkan dengan satu menghasilkan nilai atau hasil akhir 0 maka dapat dipastikan bahwa bilangan tersebut merupakan bilangan atau angka genap, maka perintah cetak print "Bilangan genap" dan ID -process nya akan di print sesuai pada perintah print kondisi if, untuk mendapatkan ID process disini saya menambahkan fungsi getpid(), sehingga masing-masing getpid() pada bilangan genap nantinya dapat dicetak atau dihasilkan dan akan ditampilkan pada output jika perintah if terpenuhi, selanjutnya saya juga membuat perintah else. Kondisi else ini digunakan apabila kondisi

pertama (if) tidak terpenuhi, artinya bilangan tersebut merupakan bilangan ganjil dan akan dilempar pada perintah else ini. Pada perintah else ini bilangan dapat dideteksi bilangan ganjil apabila bilangan $a + 1$ dan dibagi 2 hasilnya tidak 0 atau bersisa. Maka jika bilangan atau angka sesuai dengan kondisi else, perintah print atau cetak pada kondisi else akan dicetak yaitu “Bilangan Ganjil” beserta dengan ID-Processnya, dan terakhir gunakan fungsi sleep untuk menghentikan dan menjeda waktu proses. Disini saya menggunakan waktu proses 1 detik sesuai dengan perintah yang diinginkan pada brief tugasnya.

```
# Buat inputan untuk melakukan perulangan
batas=int(input("Masukkan batas perulangan: "))
```

Selanjutnya saya membuat suatu inputan angka atau bilangan dengan memanfaatkan pendeklarasian variabel dan fungsi input. Disini saya menggunakan int, karena percobaan yang dilakukan merupakan angka utuh atau bulat. Pembuatan inputan ini dilakukan agar nantinya dapat dilakukan proses perulangan proses sesuai dengan inputan yang dilakukan. Inputan disini dapat dijadikan sebagai patokan atas atau batas atas proses, misalnya jika saya memasukkan angka atau bilangan 5 maka, akan dilakukan perulangan proses sebanyak 5 kali. Jika memasukkan 3 maka akan dilakukan perulangan sebanyak 3 kali dan seterusnya.

```
for a in range(batas):
    cetak(a)
sekuensial_akhir = time()

# Multiprocessing dengan Kelas Process
process_awal = time()
print("multiprocessing.Process")
for a in range(batas):
    p=Process(target=cetak, args=(a, ))
    p.start()
    p.join()
process_akhir = time()

# Multiprocessing dengan Kelas Pool
pool_awal = time()
pool = Pool()
print("multiprocessing.Pool")
pool.map(cetak, range(0,batas))
pool.close()
pool_akhir = time()
```

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

```
# Sekuensial
sekuensial_awal = time()
print("Sekuensial")
for a in range(batas):
    cetak(a)
sekuensial_akhir = time()
```

Setelah itu langkah berikutnya adalah pembuatan 3 proses yang akan dilakukan sesuai dengan inisialisasi fungsi dan inputan perulangan tadi. Proses pertama yaitu sekuensial, langkah pertama menuliskan kode sekuensial awal dalam waktu untuk mengetahui waktu awal proses, dan perintah cetak “Sekuensial”. Langkah kedua membuat perulangan for a in range (batas)

yang berarti perulangan jika a atau bilangan pada range batas (inputan perulangan) tadi maka akan dilakukan perulangan sesuai dengan range batas perulangan (sebagai batas atas atau akhir perulangan). Setelah itu perintah cetak a dan waktu sekuensial akhirnya.

```
# Multiprocessing dengan Kelas Process
process_awal = time()
print("multiprocessing.Process")
for a in range(batas):
    p=Process(target=cetak, args=(a, ))
    p.start()
    p.join()
process_akhir = time()
```

Proses kedua adalah multiprocessing dengan kelas proses. Proses ini berbeda dengan proses sebelumnya. Pada proses ini pertama kita mendeklarasikan waktu proses awal dengan memanfaatkan fungsi `time()` untuk mengetahui waktu awal proses, lalu perintah cetak atau `print "Multiprocessing proses"`. Jika sudah dilakukan pembuatan fungsi atau kondisi perulangan dengan `for a in range (batas)` yang berarti perulangan jika a atau bilangan pada range batas (inputan perulangan) tadi maka akan dilakukan perulangan sesuai dengan range batas perulangan (sebagai batas atas atau akhir proses perulangan). Kemudian pembuatan pencetakan proses dan `args(a,)`. Hal tersebut dilakukan agar fungsi dapat menerima semua argument dan nantinya akan disimpan tanpa terkecuali dalam fungsi. Kemudian pembuatan `p.start` yang digunakan agar proses dapat dimulai dan `join` dilakukan agar proses p tidak loncat ke proses lainnya sehingga proses akan dilakukan berurut sesuai dengan inputan perulangan yang dimasukkan. Dan yang terakhir yaitu mendeklarasikan waktu akhir , agar waktu akhir proses yang sudah dilakukan dapat disimpan dengan memanfaatkan fungsi `time()` sehingga waktu akhir akan disimpan dalam fungsi tersebut.

```
# Multiprocessing dengan Kelas Pool
pool_awal = time()
pool = Pool()
print("multiprocessing.Pool")
pool.map(cetak, range(0,batas))
pool.close()
pool_akhir = time()
```

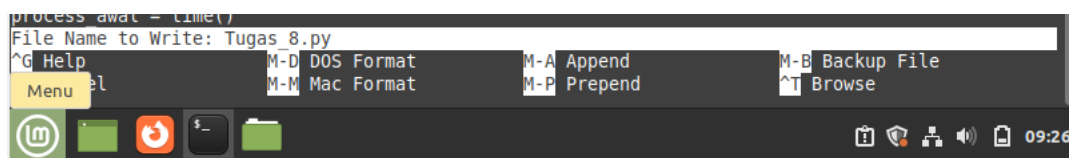
Proses ketiga adalah multiprocessing dengan kelas Pool. Pool ini berbeda dengan proses sebelumnya. Pada proses ini pertama kita mendeklarasikan waktu proses awal pool dengan memanfaatkan fungsi `time()` untuk mengetahui waktu awal proses pool dan pembuatan fungsi `Pool()` agar semua proses nantinya akan tersimpan dalam fungsi pool tersebut, lalu perintah cetak atau `print "Multiprocessing pool"`. Setelah itu pemulisan `pool.map (Cetak, range(0, batas))` yang berarti nantinya akan dilakukan proses pencetakan proses atau hasil proses multiprocessing poolnya sesuai dengan range atau perulangan yang dilakukan, yaitu dari 0 dan

berakhir sampai inputan perulangan (batas) sebagai batas akhir proses perulangan berjalan. Kemudian pendelarasian pool close () , yang dimana hal ini dilakukan agar proses pool dapat berhenti sesuai dengan perulangan dan tidak berlanjut terus menerus. Dan yang terakhir yaitu mendeklarasikan waktu akhir , agar waktu akhir proses yang sudah dilakukan dapat disimpan dengan memanfaatkan fungsi time() sehingga waktu akhir akan disimpan dalam fungsi tersebut.

```
# Membandingkan 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")
```

Setelah pembuatan ketiga proses diatas langkah terakhir yang dilakukan ialah membuat perintah cetak untuk mengetahui waktu yang diperlukan dalam ketiga proses tersebut. Dalam hal ini waktu proses dapat didapatkan dari pengurangan dari waktu proses akhir dengan waktu proses awalnya. Sehingga dari ketiga proses tersebut akan didapatkan waktu proses yang berjalan. Waktu proses ini dilakukan agar saya dapat membandingkan waktu yang diperlukan tiap-tiap atau masing-masing proses yang berbeda. Waktu ini akan dicetak dengan memanfaatkan perintah print dalam python.

5. Menyimpan Kode Script dengan CTRL + X



6. Eksekusi Program python dalam Linux

```
rangga@rangga-VirtualBox:~/Tugas 8$ python3 Tugas_8.py
```

7. Input Perulangan

```
rangga@rangga-VirtualBox:~/Tugas 8$ python3 Tugas_8.py
Masukkan batas perulangan: 3
```

8. Output Akhir

```
rangga@rangga-VirtualBox:~/Tugas 8$ python3 Tugas_8.py
Masukkan batas perulangan: 3
Sekuenial
1 Ganjil - ID proses 5589
2 Genap - ID proses 5589
3 Ganjil - ID proses 5589
multiprocessing.Process
1 Ganjil - ID proses 5590
2 Genap - ID proses 5591
3 Ganjil - ID proses 5592
multiprocessing.Pool
1 Ganjil - ID proses 5593
2 Genap - ID proses 5593
3 Ganjil - ID proses 5593
Waktu eksekusi sekuensial : 3.003281593322754 detik
Waktu eksekusi multiprocessing.Process : 3.021958351135254 detik
Waktu eksekusi multiprocessing.Pool : 3.0433509349823 detik
rangga@rangga-VirtualBox:~/Tugas 8$
```

Dari hasil atau output diatas dapat diketahui bahwa saya memasukkan inputan perulangan sebesar 3, maka proses akan dilakukan sesuai dengan perulangan yang saya masukkan yaitu sebanyak 3 proses dari tiap-tiap jenis proses (sekuensial, multiprocessing dengan proses, dan multiprocessing dengan pool). Dari ketiga proses tersebut dapat disimpulkan bahwa program akan mendeteksi dan mencetak bilangan 1 sebagai ganjil, 2 genap, dan 3 ganjil (sama semua dari ketiga proses), hanya saja ID Process yang didapatkan dan waktu proses yang diperoleh berbeda dari ketiga proses tersebut.