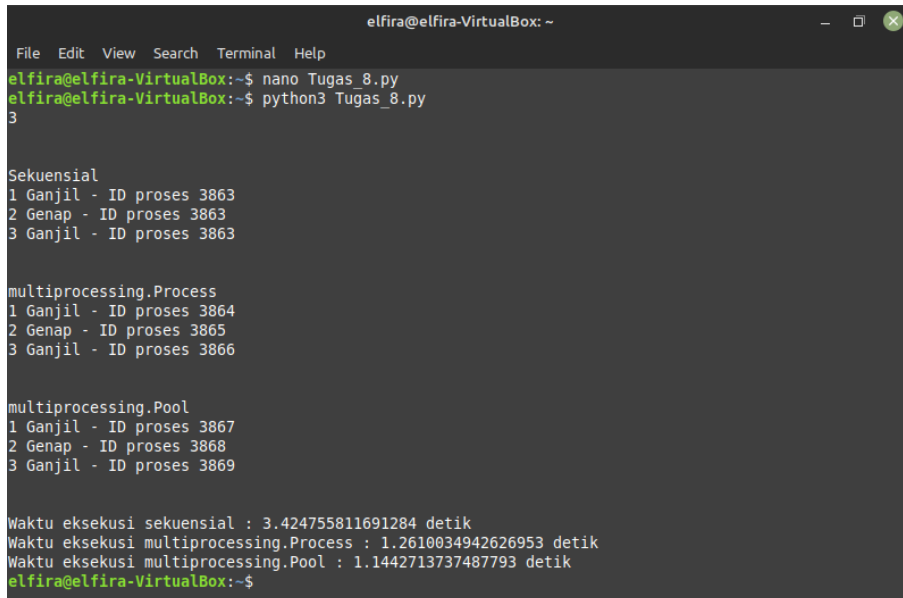


Nama : Elfira Ratna Syaharani

NPM : 21083010056

Kelas : Sistem Operasi A

Tugas 8 SISOP



```
elfira@elfira-VirtualBox: ~  
File Edit View Search Terminal Help  
elfira@elfira-VirtualBox:~$ nano Tugas_8.py  
elfira@elfira-VirtualBox:~$ python3 Tugas_8.py  
3  
  
Sekuensial  
1 Ganjil - ID proses 3863  
2 Genap - ID proses 3863  
3 Ganjil - ID proses 3863  
  
multiprocessing.Process  
1 Ganjil - ID proses 3864  
2 Genap - ID proses 3865  
3 Ganjil - ID proses 3866  
  
multiprocessing.Pool  
1 Ganjil - ID proses 3867  
2 Genap - ID proses 3868  
3 Ganjil - ID proses 3869  
  
Waktu eksekusi sekuensial : 3.424755811691284 detik  
Waktu eksekusi multiprocessing.Process : 1.2610034942626953 detik  
Waktu eksekusi multiprocessing.Pool : 1.1442713737487793 detik  
elfira@elfira-VirtualBox:~$
```

Keterangan:

- 1) Nano Tugas_8.py digunakan untuk membuat file baru atau membuat script untuk laporan tugas 8.
- 2) Python3 digunakan untuk mengeksekusi file py yang sebelumnya telah dibuat. Output yang dihasilkan adalah
 - a. Menginput angka 3 sebagai x.
 - b. Karena x adalah 3, maka range pada pemrosesan otomatis juga 3 yang artinya hanya akan memproses sebanyak 3 kali. Kemudian bilangan ganjil dan genap dapat ditentukan dengan modulo atau %. Angka 1 dan 3 merupakan ganjil karena $1 \bmod 2 = 1$ dan $3 \bmod 2 = 1$, sedangkan $2 \bmod 2 = 0$ dan apabila $i \% 2 == 0$ maka termasuk genap.
 - c. ID proses pada sekuensial memiliki nilai yang sama, karena akan dilakukan 3 pemrosesan maka ID process diakhiri angka 3.
 - d. ID proses pada multiprocessing.Process memiliki nilai-nilai yang berbeda dan secara beruntun.
 - e. ID proses pada multiprocessing.Pool memiliki nilai-nilai yang berbeda dan secara beruntun namun disesuaikan dengan jumlah CPU yang pada komputer.
 - f. Dapat diketahui bahwa waktu eksekusi pada sekuensial lebih lambat dibandingkan dengan waktu eksekusi pada multiprocessing.Process dan multiprocessing.Pool.

```
elfira@elfira-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 6.2 Tugas_8.py
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

x = int(input())
def cetak(i):
    print(i+1, ("Ganjil" if (i%2==0) else "Genap"), "- ID proses", getpid())
    sleep(1)
    print("\n")

print("Sekuensial")
sekuensial_awal = time()
for i in range(x):
    cetak(i)
sekuensial_akhir = time()
print("\n")

print("multiprocessing.Process")
kumpulan_proses = []
process_awal = time()
for i in range(x):
    p = Process(target=cetak, args=(i,))
    kumpulan_proses.append(p)
    p.start()
for i in kumpulan_proses:
    p.join()
process_akhir = time()
print("\n")

print("multiprocessing.Pool")
pool_awal = time()
pool = Pool()
pool.map(cetak, range(0,x))
pool.close()
pool_akhir = time()
print("\n")

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")

^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
```

Keterangan:

- 1) Mengimport built-in libraries yang akan digunakan seperti:
 - a. Getpid yang digunakan untuk mengambil ID proses.
 - b. Time yang digunakan untuk mengambil waktu dalam bentuk detik.
 - c. Sleep yang digunakan untuk memberi jeda waktu dalam bentuk detik.
 - d. Pool yang digunakan untuk sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer.
 - e. Process adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.
- 2) Membuat x agar berisi angka inputan yang akan dipakai sebagai range menggunakan `int(input())`.
- 3) Mendeklarasikan fungsi cetak yang akan mencetak angka dari variabel i, tulisan ganjil genap yang ditentukan dengan modulo atau apabila variabel i modulo bernilai 0 maka bilangan tersebut adalah genap, namun apabila bernilai selain 0 atau bernilai 1 maka bilangan tersebut adalah ganjil. Kemudian tulisan "- ID proses" dan menggunakan `getpid()` untuk mengetahui ID proses tersebut.
- 4) `Print("\n")` digunakan untuk memberi jarak baris antar output.
- 5) `Print("Sekuensial")` digunakan untuk membuat teks Sekuensial.

- 6) `Sekuensial_awal=time()` dan `Sekuensial_akhir=time()` digunakan untuk mendapatkan waktu sebelum dan sesudah eksekusi dari pemrosesan sekuensial.
- 7) `For i in range(x): cetak(i)`, digunakan sebagai proses berlangsung, dimana banyaknya pemrosesan menggunakan `range` dan `x` yang akan diinputkan.
- 8) `Print("multiprocessing.Process")` digunakan untuk membuat tulisan `multiprocessing.Process`.
- 9) `Process_awal=time()` dan `Process_akhir=time()` digunakan untuk mendapatkan waktu sebelum dan sesudah eksekusi dari pemrosesan `multiprocessing.Process`.
- 10) Fungsi `p.start()` digunakan untuk memulai proses dan ketika memanggil `p.join` akan menghentikan proses.
- 11) Proses-proses ditambahkan ke dalam antrian dengan menggunakan `kumpulan_proses.append(p)`.
- 12) `Print("multiprocessing.Pool")` digunakan untuk membuat teks `multiprocessing.Pool`.
- 13) `Pool_awal=time()` dan `Pool_akhir=time()` digunakan untuk mendapatkan waktu sebelum dan sesudah eksekusi dari pemrosesan `multiprocessing.Pool`.
- 14) `Pool.map()` mengambil fungsi yang ingin diparalelkan dan dapat diulang sebagai argumen.
- 15) `Pool.close()` biasanya dipanggil ketika bagian paralel dari program utama telah selesai. Maka proses akan berakhir ketika semua yang sudah ditugaskan telah selesai.
- 16) Langkah terakhir yaitu membandingkan waktu eksekusi dari tiga jenis pemrosesan tersebut. Dimana rumus untuk mengetahui waktu eksekusi adalah waktu akhir dikurangi dengan waktu awal dan menggunakan satuan detik.