LAPORAN 8 MULTIPROCESSING

SISTEM OPERASI KELAS A

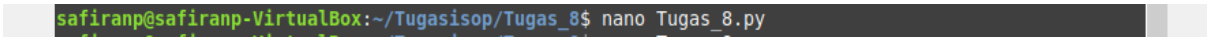
SAFIRA NAJMA PRAMISWARI

21083010074

SAINS DATA

MULTIPROCESSING LINUX

1. Pertama, seperti biasa membuat file dengan nano namafile.formatfile.
   Nah, karena disini kita mau ngoding dan disini kita mau gunakanpython maka kita gunakan “nano namafile.py”

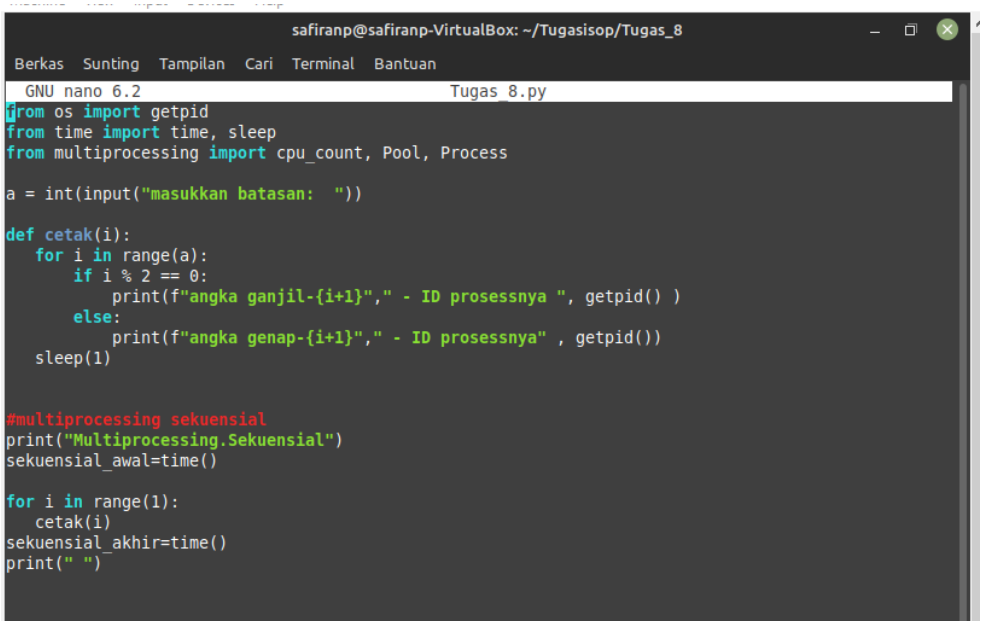


2. **Multiprocessing** berkaitan dengan potensi sistem yang mendukung lebih dari satu prosesor pada satu waktu. Semua aplikasi dalam multiprocessing dihancurkan menjadi pola-pola kecil, dan masing-masing pola tersebut bergantung pada dirinya sendiri dalam bekerja. Dalam jenis proses ini, dua atau lebih proses digunakan. Mari kita pertimbangkan bahwa komputer berjalan dengan menggunakan prosesor tunggal. Jika kami menetapkan beberapa prosesor sekaligus, itu akan menyebarkan setiap tugas beralih ke tugas lain setelah waktu yang singkat untuk menjaga semua proses tetap berjalan. Dengan kata lain, jika ada lebih banyak tugas yang harus dilakukan, lebih sulit, itu adalah mempertahankannya di jalur penyelesaian. Sedangkan mengalokasikan setiap tugas waktu yang sama merupakan tantangan besar juga. Dalam kondisi ini, kita perlu memunculkan konsep multiprocessing. Sebuah sistem multiprocessing dapat memungkinkan.
   Nah, pertama kita ketik terlebih dahulu library apa saja yang kita gunakan
   - from os import getpid = Dalam setiap proses, “PID” diperoleh melalui fungsi get(). Kami telah menggunakan dua modul. Salah satunya adalah multiprocessing, dan yang lainnya adalah sistem operasi OS.
   - from time import time, sleep = untuk memberi jeda waktu(detik)
   - from multiprocessing import cpu_count, Pool, Process = untuk mengeksekusi proses dengan banyak input dan mendistribusikan data input ini di antara proses.

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

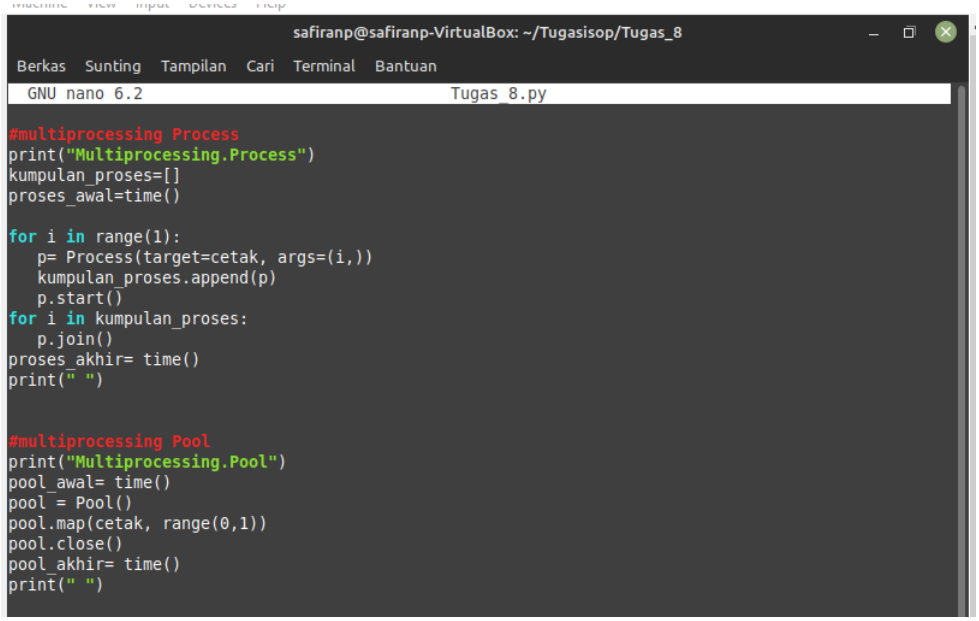
Disini kita pakai fungsi sebagai berikut

```
def cetak(i):
   for i in range(a):
       if i % 2 == 0
           print(f”angka ganjil{i+1}”,” – ID prosessnya “, getpid() )
       else:
           print(f”angka genap{i+1}”,” – ID prosessnya “, getpid())
   sleep(1)
```



Di multiprocessing, proses muncul dengan membuat objek Proses dan kemudian memanggil metode start() nya. Proses mengikuti API dari threading.Thread. Contoh sepele dari program multiproses adalah Pada contoh gambar dibawah,

Dalam contoh ini, kita akan melihat cara kerja multiprocessing dan import time, pool, cpu_count. POOL digunakan untuk mengeksekusi proses dengan banyak input dan mendistribusikan data input ini di antara proses. **Fungsi pool** memberikan nilai dan memanggil fungsi secara bersamaan. Maka kita perlu menghentikan timer. Fungsi pengatur waktu ini akan menghitung detik yang digunakan untuk fungsi ini, yang dikenal sebagai waktu yang telah berlalu; itu diperoleh dengan mengurangkan nilai awal waktu dari nilai akhir.



Dalam komputer berprosesor tunggal, hanya satu instruksi yang dapat bekerja dalam satu waktu sehingga CPU tersebut secara aktif mengolah instruksi untuk satu pekerjaan tersebut. Multitasking mengatasi masalah ini dengan menjadwalkan pekerjaan mana yang dapat berjalan dalam satu waktu, dan kapan pekerjaan lain yang menunggu untuk diolah dapat dikerjakan.

```
#waktu eksekusi
print("Waktu eksesusi Multiprocessing.Sekuensial: ",sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi Multiprocessing.Process    : ",proses_akhir - proses_awal, "detik")
print("Waktu ekseskusi Multiprocessing.Pool      : ",pool_akhir - pool_awal, "detik")

^G Bantuan      ^O Tulis       ^W Cari        ^K Potong      ^T Jalankan    ^C Lokasi
^X Keluar       ^R Baca        ^\ Ganti       ^U Tempel      ^J Ratakan     ^/ Ke Baris
```

3. Terakhir, setelah kita ngoding dengan script dalam tab linux sesuai yang kita buat, kita panggil ourput sesuai bentuk file. Karena formatfile yang kita gunakan python maka kita panggil output dengan "python3 namafile.py"

```
safiranp@safiranp-VirtualBox:~/Tugasisop/Tugas_8$ python3 Tugas_8.py
```

Sistem operasi komputer dapat mengadopsi berbagai macam penjadwalan, yaitu:

1. Dalam sistem Multi-program, pekerjaan yang sedang diolah terus berjalan hingga membutuhkan suatu operasi yang memerlukan interaksi dari luar. Multi-program dipakai untuk memaksimalkan penggunaan CPU.
2. Dalam sistem Time-sharing, pekerjaan yang sedang diolah diharuskan melepaskan kerja CPU. Sistem time-sharing didesain untuk memperbolehkan beberapa program seolah diproses secara bersamaan.
3. Dalam sistem Real-time, beberapa program yang sedang menunggu dijamin untuk mendapatkan pengolahan dari CPU ketika interaksi luar terjadi.

Dibawah ini merupakan Output dari code Multiprocessing dengan kita masukkan Batasan sebesar 3 yang sesuai dengan latihan soal 8.

Disimpulkan jika proses sekuensial lebih lambat dibanding multiprocessing namun bukan berarti kita harus melakukan multiprocessing terus menerus, gunakan metode sesuai kebutuhan. Dapat diilihat pada gambar dibawah ini,

```
safiranp@safiranp-VirtualBox:~/Tugasisop/Tugas_8$ python3 Tugas_8.py
masukkan batasan:  3
Multiprocessing.Sekuensial
angka ganjil-1  - ID prosessnya  2300
angka genap-2  - ID prosessnya 2300
angka ganjil-3  - ID prosessnya  2300

Multiprocessing.Process
angka ganjil-1  - ID prosessnya  2301
angka genap-2  - ID prosessnya 2301
angka ganjil-3  - ID prosessnya  2301

Multiprocessing.Pool
angka ganjil-1  - ID prosessnya  2302
angka genap-2  - ID prosessnya 2302
angka ganjil-3  - ID prosessnya  2302

Waktu eksesusi Multiprocessing.Sekuensial:  1.0014817714691162 detik
Waktu eksekusi Multiprocessing.Process    :  1.052286148071289 detik
Waktu ekseskusi Multiprocessing.Pool      :  1.1109974384307861 detik
safiranp@safiranp-VirtualBox:~/Tugasisop/Tugas_8$
```