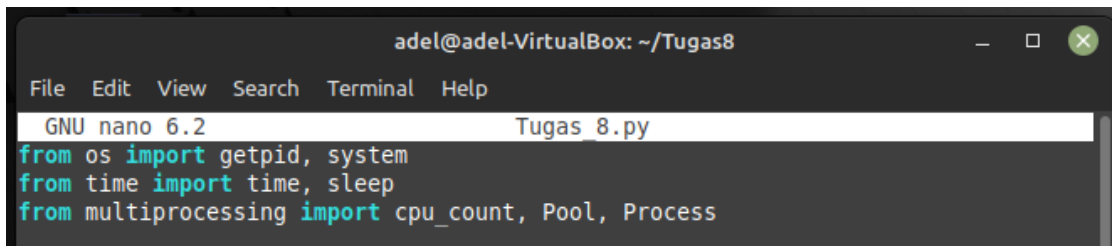


LAPORAN TUGAS 8

A. Import library yang dibutuhkan

A screenshot of a terminal window titled 'adel@adel-VirtualBox: ~/Tugas8'. The window shows the GNU nano 6.2 editor with the file 'Tugas 8.py'. The code visible is:

```
from os import getpid, system
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

getpid digunakan untuk mengambil ID proses

time digunakan untuk mengambil waktu(detik)

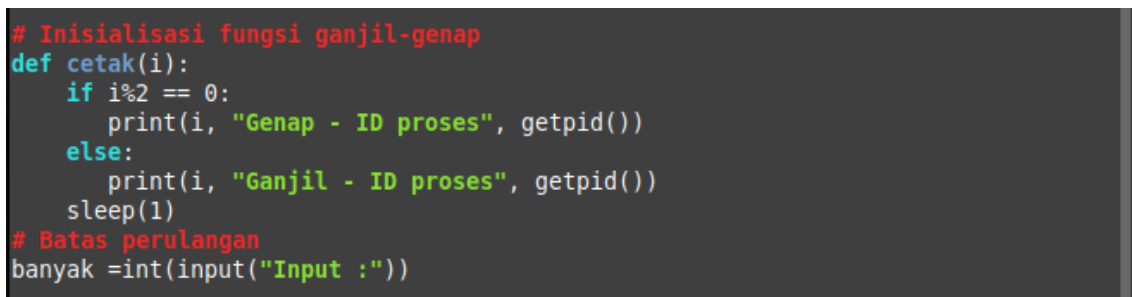
sleep digunakan untuk memberi jeda waktu(detik)

cpu_count digunakan untuk melihat jumlah CPU

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 computer

B. Inisialisasi fungsi ganjil genap dan batas perulangan

A screenshot of a terminal window showing the initialization of the odd-even function and the loop limit. The code is:

```
# Inisialisasi fungsi ganjil-genap
def cetak(i):
    if i%2 == 0:
        print(i, "Genap - ID proses", getpid())
    else:
        print(i, "Ganjil - ID proses", getpid())
        sleep(1)
# Batas perulangan
banyak =int(input("Input :"))
```

Fungsi cetak pada def cetak (i) :

- Untuk nilai *i* genap atau pembagian nilai *i* dengan 2 tidak bersisa, akan mencetak **nilai i Genap – ID proses nilai ID proses**
- Untuk nilai *i* ganjil atau pembagian nilai *i* dengan 2 memiliki sisa, maka akan mencetak **nilai i Ganjil – ID proses nilai ID proses**
- fungsi sleep untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan.

Mendeklarasikan variabel **banyak** sebagai batas perulangan yang nilainya adalah hasil input user.

C. Pemrosesan sekuensial

```
# Sekuensial
print("Sekuensial")
# Untuk mendapatkan waktu sebelum eksekusi
sekuensial_awal = time()
# Proses berlangsung
for i in range(1, banyak+1):
    cetak(i)
# Untuk mendapatkan waktu setelah eksekusi
sekuensial_akhir = time()
```

D. Multiprocessing dengan kelas process

```
# Process
print("multiprocessing.Process")
# Untuk menampung proses-proses
kumpulan_proses = []
# Untuk mendapatkan waktu sebelum eksekusi
process_awal = time()
# Proses berlangsung
for i in range(1, banyak+1):
    p = Process(target=cetak, args=(i,))
    kumpulan_proses.append(p)
    p.start()
# Untuk menggabungkan proses-proses agar tidak loncat ke proses sebelumnya
for i in kumpulan_proses:
    p.join()
# Untuk mendapatkan waktu setelah eksekusi
process_akhir = time()
```

E. Multiprocessing dengan kelas pool

```
# Pool
print("multiprocessing.Pool")
# Untuk mendapatkan waktu sebelum eksekusi
pool_awal = time()
# Proses berlangsung
pool = Pool()
pool.map(cetak, range(1, banyak+1))
pool.close()
# Untuk mendapatkan waktu setelah eksekusi
pool_akhir = time()
```

F. Membandingkan waktu eksekusi

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

OUTPUT

```
adel@adel-VirtualBox:~/Tugas8$ python3 Tugas_8.py
Input :3
Sekuensial
1 Ganjil - ID proses 1743
2 Genap - ID proses 1743
3 Ganjil - ID proses 1743
multiprocessing.Process
1 Ganjil - ID proses 1744
2 Genap - ID proses 1745
3 Ganjil - ID proses 1746
multiprocessing.Pool
1 Ganjil - ID proses 1747
2 Genap - ID proses 1747
3 Ganjil - ID proses 1747
Sekuensial : 3.003903388977051 detik
Kelas Process : 1.0451180934906006 detik
Kelas Pool : 3.0718443393707275 detik
```

Hasil akhir dari output menunjukkan bahwa multiprocessing dengan kelas process adalah proses paling cepat dibandingkan pemrosesan sekuensial dan multiprocessing dengan kelas pool.