

LAPORAN SISTEM OPERASI WEEK-8

Nama : Novita Anggraini
NPM : 21083010104
Mata Kuliah : Sistem Operasi A

Multiprocessing

Multiprocessing mengacu pada kemampuan sistem untuk mendukung lebih dari satu prosesor pada saat yang sama. Aplikasi dalam sistem multiprosesor dipecah menjadi rutinitas yang lebih kecil yang berjalan secara independen. Sistem operasi mengalokasikan utas (thread) ke prosesor untuk meningkatkan kinerja sistem.

Soal Latihan

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

Batasan:

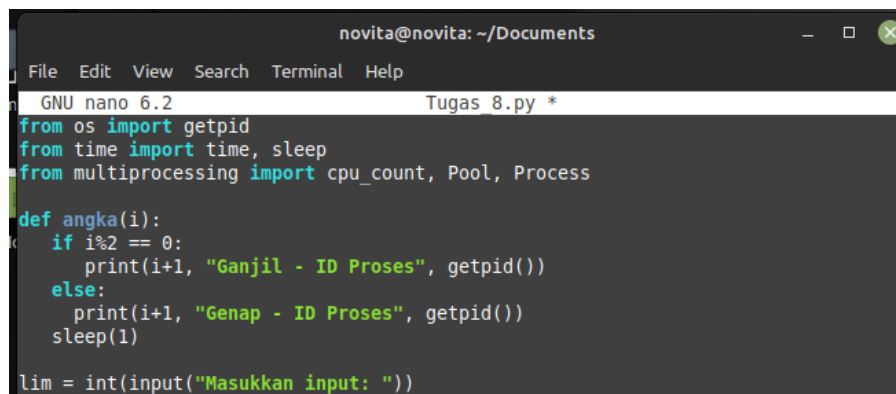
- Nilai yang dijadikan argumen pada fungsi sleep() adalah satu detik.
- Masukkan jumlahnya satu dan berupa bilangan bulat.
- Masukkan adalah batas dari perulangan tersebut.
- Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan paralel.

Penjelasan

Tugas ini diselesaikan dengan menggunakan bahasa pemrograman Python yang dijalankan pada sistem operasi Linux.

```
novita@novita:~/Documents$ nano Tugas_8.py
```

- Langkah pertama adalah dengan menjalankan command nano Tugas_8.py untuk membuka text editor.
- Setelah text editor terbuka, maka kita dapat mulai menuliskan syntax Python untuk multiprocessing.



```
novita@novita: ~/Documents
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

def angka(i):
    if i%2 == 0:
        print(i+1, "Ganjil - ID Proses", getpid())
    else:
        print(i+1, "Genap - ID Proses", getpid())
        sleep(1)

lim = int(input("Masukkan input: "))
```

```

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

def angka(i):
    if i%2 == 0:
        print(i+1, "Ganjil - ID Proses", getpid())
    else:
        print(i+1, "Genap - ID Proses", getpid())
        sleep(1)

lim = int(input("Masukkan input: "))

#Sekuensial
first_seq = time()
print("\n Sekuensial")
for i in range(lim):
    angka(i)
last_seq = time()

#multiprocessing.Process
print("\n multiprocessing.Process")
array_proses = []
first_process = time()
for i in range(lim):
    proses = Process(target=angka, args=(i, ))
    array_proses.append(proses)
    proses.start()
for i in array_proses:
    proses.join()
last_process = time()

#multiprocessing.Pool
print("\n multiprocessing.Pool")
first_pool = time()
pool = Pool()
pool.map(angka, range(0, lim))
pool.close()
last_pool = time()

#Perbandingan
total_seq = last_seq - first_seq
total_process = last_process - first_process
total_pool = last_pool - first_pool

print("\nWaktu eksekusi sekuensial: ", total_seq, "detik")
print("Waktu eksekusi multiprocessing.Process: ", total_process, "detik")
print("Waktu eksekusi multiprocessing.Pool:" , total_pool, "detik")

```

Steps:

1. Import Library

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

Modul OS merupakan module pada python untuk program python agar python berinteraksi langsung dengan sistem operasi yang digunakan, sedangkan getpid digunakan untuk mengambil ID proses. Library time digunakan untuk mengambil waktu dalam detik, sleep digunakan untuk memberi jeda waktu pada proses yang mana juga dalam bentuk detik. cpu_count digunakan untuk melihat jumlah CPU yang beroperasi pada komputer. Pool digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer. Lalu Process adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.

2. Mendefinisikan 'cetak'/angka yang diproses

```
def angka(i):
    if i%2 == 0:
        print(i+1, "Ganjil - ID Proses", getpid())
    else:
        print(i+1, "Genap - ID Proses", getpid())
    sleep(1)
```

Dalam soal latihan ini, akan terdapat angka yang dimasukkan menjadi input. Function angka di sini digunakan untuk memproses sesuai dengan apa yang diminta oleh soal, yakni untuk mengetahui apakah input tersebut bernilai ganjil atau genap.

Angka ganjil dan genap dapat ditandai dengan modulo (%) atau sisa bagi. Jika suatu angka dibagi dengan 2 menyisakan sisa 0, maka angka tersebut dipastikan angka genap. Selain itu, yang mana artinya menyisakan angka, maka dianggap sebagai angka ganjil. Dilanjutkan dengan menuliskan fungsi sleep dengan nilai 1 yang artinya akan terdapat jeda waktu satu detik selama pemrosesan.

3. Memasukkan nilai limit/batas

```
lim = int(input("Masukkan input: "))
```

Proses akan dilakukan sebanyak nilai yang diinputkan. Jika nilai yang diinputkan '3', maka proses akan dilakukan sebanyak 3 kali (1, 2, 3). Nilai ini adalah sebagai nilai batas atas atau limit pemrosesan.

4. Sekuensial

```
#Sekuensial
first_seq = time()
print("\n Sekuensial")
for i in range(lim):
    angka(i)
last_seq = time()
```

Dalam multiprocessing model ini, kita buat timestamp waktu mulai dalam variabel 'first_seq', dan 'last_seq' untuk timestamp waktu ketika proses berakhir. Proses yang dimaksud di sini adalah penentuan sebuah angka, dalam hal ini 1 hingga batas limit, apakah angka tersebut termasuk ganjil atau genap. Yang mana untuk memproses penentuan ganjil genap itu sudah terekam dalam fungsi 'angka'. Kita hanya perlu menuliskan nama fungsi diikuti dengan variabel yang mengandung angka.

5. Multiprocessing proses

```
#multiprocessing.Process
print("\n multiprocessing.Process")
array_proses = []
first_process = time()
for i in range(lim):
    proses = Process(target=angka, args=(i, ))
    array_proses.append(proses)
    proses.start()
for i in array_proses:
    proses.join()
last_process = time()
```

Konsep yang digunakan di bagian ini adalah penggunaan array sebagai penampung proses. Variabel first_process digunakan untuk menyimpan timestamp mulai, dan variabel last_process digunakan untuk menyimpan timestamp berakhir. Dalam multiprocessing, proses dimunculkan dengan membuat objek Process kemudian memanggil start() dalam metodenya. Tugas dapat dijalankan dalam proses baru dengan membuat instance dari kelas proses dan menentukan fungsi untuk dijalankan dalam proses baru melalui argumen " target ". Proses dilakukan satu per satu, dan nilai yang dihasilkan akan disimpan dalam array.

6. Multiprocessing Pool

```
#multiprocessing.Pool
print("\n multiprocessing.Pool")
first_pool = time()
pool = Pool()
pool.map(angka, range(0, lim))
pool.close()
last_pool = time()
```

Kelas kumpulan multiproses Python memungkinkan kita untuk membuat dan mengelola kumpulan proses di Python. Python menyediakan kumpulan proses melalui kelas `multiprocessing.pool`. Dengan menggunakan `multiprocessing.pool` dapat memungkinkan tugas diserahkan sebagai fungsi ke kumpulan proses untuk dieksekusi secara bersamaan. Timestamp mulainya proses disimpan dalam variabel `first_pool`, dan timestamp berakhirnya proses disimpan dalam variabel `last_pool`.

7. Perbandingan

```
#Perbandingan
total_seq = last_seq - first_seq
total_process = last_process - first_process
total_pool = last_pool - first_pool
print("\nWaktu eksekusi sekuensial: ", total_seq, "detik")
print("Waktu eksekusi multiprocessing.Process: ", total_process,
      "detik")
print("Waktu eksekusi multiprocessing.Pool:" , total_pool, "detik")
```

Setelah tiga proses berjalan, maka kita telah memiliki 6 data waktu proses, yakni:

1. `first_seq` : waktu mulai sekuensial
2. `last_seq` : waktu berakhir sekuensial
3. `first_process` : waktu mulai `multiprocessing.Process`
4. `last_process` : waktu berakhir `multiprocessing.Process`
5. `first_pool` : waktu mulai `multiprocessing.Pool`
6. `last_pool` : waktu berakhir `multiprocessing.Pool`

Maka untuk membuat perbandingan, kita perlu mencari selisih waktu mulai dan waktu berakhir setiap proses.

Script di atas telah disimpan di dalam file `Tugas_8.py`. Untuk mengeksekusinya dalam Linux, maka kita gunakan command “`python3 Tugas_8.py`”. Output yang dihasilkan adalah sebagai berikut:

```
novita@novita:~/Documents$ python3 Tugas_8.py
Masukkan input: 3

Sekuensial
1 Ganjil - ID Proses 25404
2 Genap - ID Proses 25404
3 Ganjil - ID Proses 25404

multiprocessing.Process
1 Ganjil - ID Proses 25406
3 Ganjil - ID Proses 25408
2 Genap - ID Proses 25407

multiprocessing.Pool
1 Ganjil - ID Proses 25409
2 Genap - ID Proses 25410
3 Ganjil - ID Proses 25410

Waktu eksekusi sekuensial: 3.0033130645751953 detik
Waktu eksekusi multiprocessing.Process: 1.0360307693481445 detik
Waktu eksekusi multiprocessing.Pool: 2.0853683948516846 detik
```

Sehingga dapat disimpulkan proses multiprocessing tercepat adalah dengan metode kedua.