

REPORT ASSIGNMENT
OPERATING SYSTEM A CLASS



“ Multiprocessing ”

Name :

Nurmalita Fitri Ramadani (21083010067)

Lecturer :

Mohammad Idhom, S.P., S.Kom., M.T.

Nine Alvariqati (Assistant Lecturer)

DEPARTMENT OF DATA SCIENCE
COMPUTER SCIENCE FACULTY
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2022

Code :

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

#Function Declaration
i=input("Masukkan bilangan\n")
i=int(i)
def hitung(i):
    if (i%2) == 0:
        print(i, "Genap - ID Process", getpid())
    else:
        print(i, "Ganjil - ID Process", getpid())
    sleep(1)

#Sequential Process
print ("\nSekuensial")
sekuensial_awal = time()

for j in range(1, i+1):
    hitung(j)

sekuensial_akhir = time()

#multiprocess Class
print ("\nmultiprocessing.Process")

kumpulan_proses = []
proses_awal = time()

for k in range(1, i+1):
    p = Process(target=hitung, args=(k,))
    kumpulan_proses.append(p)
    p.start()

for k in kumpulan_proses:
    p.join()

proses_akhir = time()

#Pool
print ("\nmultiprocessing.Pool")

pool_awal = time()
pool = Pool()
pool.map(hitung, range(1, i+1))
pool.close()

pool_akhir = time()
```

```
#Time Compare
print ("\nWaktu eksekusi sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print ("Waktu eksekusi multiprocessing.Process :", proses_akhir - proses_awal, "detik")
print ("Waktu eksekusi multiprocessing.Pool :", pool_akhir - pool_awal, "detik")
```

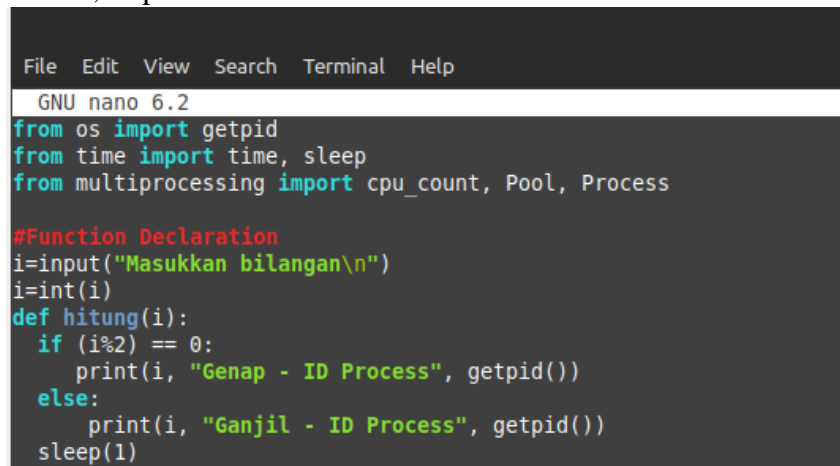
Task! :

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

Terms and Condition:

- 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.

1. First step. Create, import and Function Declaration



```
File Edit View Search Terminal Help
GNU nano 6.2
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

#Function Declaration
i=input("Masukkan bilangan\n")
i=int(i)
def hitung(i):
    if (i%2) == 0:
        print(i, "Genap - ID Process", getpid())
    else:
        print(i, "Ganjil - ID Process", getpid())
    sleep(1)
```

- Create the file with nano command and put .py at the end of the file name so that we can access it as python file
- After creating the file, write the necessary library for the process such as:
 - a) os: import the getpid function that used for returns the process ID (PID) of the calling process.
 - b) time: import time function that used for print out the taken time of the process, sleep for pause (second).
 - c) multiprocessing: cpu_count used for see the cpu count, Pool used for parallel execution of a function across multiple input values, distributing the input data across processes. Last one is Process that used for perform parallel processing by using process successively on the computer.

- Decraling the function that will print out the number type information and ID Process using if else. The (i%1)%2 is for calculating is the number is an odd or even.

2. Second, Write code for the Sequential Process

```
#Sequential Process
print ("\nSekuensial")
sekuensial_awal = time()

for j in range(1, i+1):
    hitung(j)

sekuensial_akhir = time()
```

- Sekuensial_awal = time, is a variable that will be used for telling the taking time of the process in the beginning.
- Sekuensial_akhir = time, is a variable that will be used for telling the taking time of the end of the process.
- Looping for in range and call the hitung function for the calculation.

3. Third, Write class for Multiprocessing Process.

```
print ("\nmultiprocessing.Process")

kumpulan_proses = []
proses_awal = time()

for k in range(1, i+1):
    p = Process(target=hitung, args=(k,))
    kumpulan_proses.append(p)
    p.start()

for l in kumpulan_proses:
    p.join()

proses_akhir = time()
```

- Kumpulan_proses = [], is a variable for collect all class that written for the process.
- Proses_awal = time(), is a variable is a variable that will be used for telling the taking time of the Multiprocessing process in the beginning.
- p.join() in looping for in range is used for combine processes so as not to jump to processes previously.
- Proses_akhir = time(), is a variable that will be used for telling the taking time of the end of the Multiprocessing process.

4. Forth, Write for Multiprocessing Pool

```
#Multiprocess Pool
print ("\nmultiprocessing.Pool")

pool_awal = time()
pool = Pool()
pool.map(hitung, range(1, i+1))
pool.close()

pool_akhir = time()
```

- Pool_awal = time(), is a variable is a variable that will be used for telling the taking time of the Multiprocessing pool in the beginning.
- pool.map(), used to call function calls each available CPU 0-n times. For n is an input constraint from user.
- Pool_akhir = time(), is a variable that will be used for telling the taking time of the end of the Multiprocessing pool.

5. Fifth, Call all the syntax

```
#Time Compare
print ("\nWaktu eksekusi sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print ("Waktu eksekusi multiprocessing.Process :", proses_akhir - proses_awal, "detik")
print ("Waktu eksekusi multiprocessing.Pool :", pool_akhir - pool_awal, "detik")
```

Use print command to write the title of the output. Call the related or necessary syntax for that title to show what we want to make.

6. Last, use python3 Tugas_8.py to see the output

```
litaa@litaa-VirtualBox:~/week8$ python3 Tugas_8.py
Masukkan bilangan
4

Sekuensial
1 Ganjil - ID Process 2526
2 Genap - ID Process 2526
3 Ganjil - ID Process 2526
4 Genap - ID Process 2526

multiprocessing.Process
1 Ganjil - ID Process 2527
2 Genap - ID Process 2528
3 Ganjil - ID Process 2529
4 Genap - ID Process 2530

multiprocessing.Pool
1 Ganjil - ID Process 2531
2 Genap - ID Process 2531
3 Ganjil - ID Process 2531
4 Genap - ID Process 2531

Waktu eksekusi sekuensial : 4.0048394203186035 detik
Waktu eksekusi multiprocessing.Process : 1.059741735458374 detik
Waktu eksekusi multiprocessing.Pool : 4.0637125968933105 detik
```