

TUGAS PEMROGRAMAN JARINGAN

CONCURRENCY



Oleh:

Putu Putri Natih Devayanti

05111840000163

Kelas Pemrograman Jaringan D

Departemen Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember (ITS) Surabaya

2021

Tugas

1. Buatlah program yang mengimplementasikan

- Multi process
- Multi thread
- Multi process asynchronous
- Multi thread asynchronous

Dengan menggunakan protokol transport UDP, kasus dapat didefinisikan sendiri dan buatlah arsitektur jaringan anda sendiri di simulator gns3

2. Buatlah laporan dalam bentuk PDF yang berisikan screenshot dari

- deskripsi kasus yang dibuat
- gambar arsitektur jaringan (dalam simulator GNS3)
- program yang dibuat (1-4)
- hasil outputnya

Jawab

- Kasus yang digunakan pada implementasi concurrency kali ini adalah mengunduh sebuah file bertipe gambar dengan project GNS3 yang terdiri dari 3 alpine dengan spesifikasi peran dan IP address sebagai berikut.

- Server :

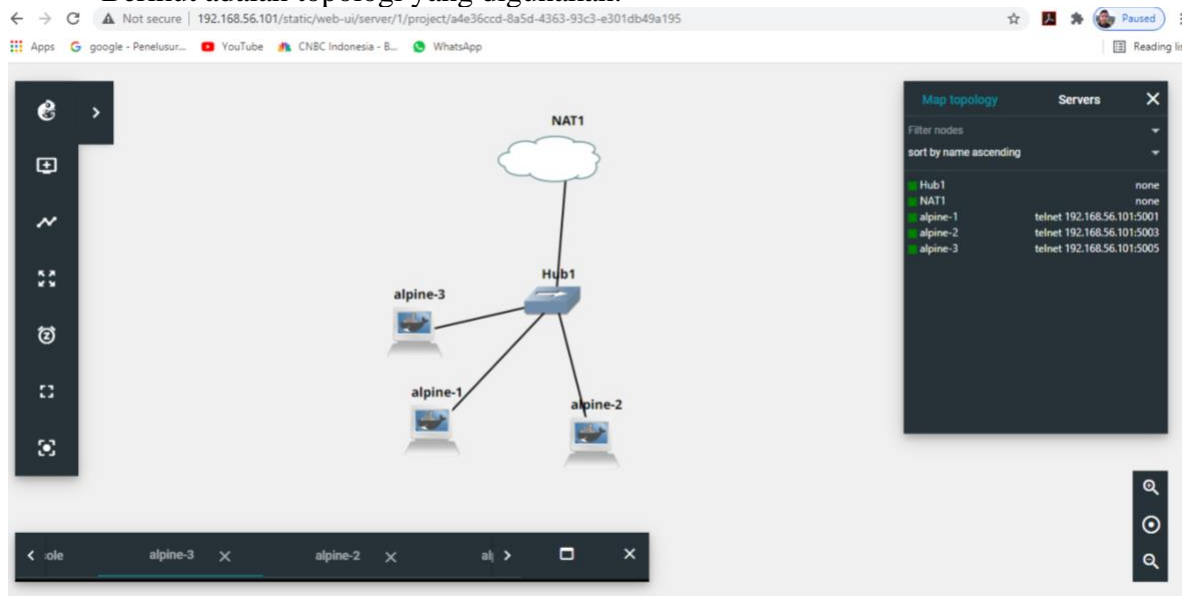
alpine-2 : 192.168.122.131 (didapatkan dari inconfig pada alpine)

alpine-3 : 192.168.122.121 (didapatkan dari inconfig pada alpine)

- Client :

alpine-1 : 192.168.122.118 (didapatkan dari inconfig pada alpine)

Berikut adalah topologi yang digunakan.



Langkah-langkah pengerjaannya adalah sebagai berikut.

- Buat file_server1.py dan file_server2.py pada folder Pemrograman_Jaringan_D/progjar3/Jawaban3

- Ubah UDP_IP_ADDRESS pada file_server1.py menjadi IP Address dari alpine-2

```

GNU nano 4.6 file_server1.py
import socket

UDP_IP_ADDRESS = '192.168.122.131'
UDP_PORT = 5758

serverSock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
serverSock.bind((UDP_IP_ADDRESS,UDP_PORT))
filename='server1.jpg'
fp = open(filename,'wb+')
ditulis=0
count=0
while True:
    data, addr = serverSock.recvfrom(1024)
    count=count+len(data)
    print(addr, count,len(data), data)
    fp.write(data)

[ Wrote 16 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Paste Text ^T To Spell  ^_ Go To Line

```

- Ubah UDP_IP_ADDRESS pada file_server2.py menjadi IP Address dari alpine-3

```

GNU nano 4.6 file_server2.py
import socket

UDP_IP_ADDRESS = '192.168.122.121'
UDP_PORT = 5758

serverSock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
serverSock.bind((UDP_IP_ADDRESS,UDP_PORT))
filename='server2.jpg'
fp = open(filename,'wb+')
ditulis=0
count=0
while True:
    data, addr = serverSock.recvfrom(1024)
    count=count+len(data)
    print(addr, count,len(data), data)
    fp.write(data)

[ Read 16 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Paste Text ^T To Spell  ^_ Go To Line

```

- Kemudian lakukan modifikasi pada file-file yang ada pada progjar3 berikut.

- File library.py

```

1  import logging
2  import requests
3  import socket
4  import os
5  import time
6  import datetime

7
8  def get_url_list():
9      urls = dict()
10     urls['olivia'] = 'https://m.media-amazon.com/images/M/MV5BYTYxODRmNzI1MGFhNS00MzI4LTk0NTM1YjdiYjgwYmNhODQ2XkEyXkFqcGdeQXVyMTg1ME'
11     urls['taylor'] = 'http://iad.iberita.com/wp-content/uploads/2014/06/taylor-swift-presenting-jpg.jpg'
12     return urls
13
14  def download_gambar(url=None, tuliskefile='image'):
15     waktu_awal = datetime.datetime.now()
16     if url is None:
17         return False
18     ff = requests.get(url)
19     tipe = dict()
20     tipe['image/png'] = 'png'
21     tipe['image/jpg'] = 'jpg'
22     tipe['image/gif'] = 'gif'
23     tipe['image/jpeg'] = 'jpg'
24     tipe['application/zip'] = 'zip'
25     tipe['video/quicktime'] = 'mov'
26     # time.sleep(2) # untuk simulasi, diberi tambahan delay 2 detik
27
28     content_type = ff.headers['Content-Type']
29     logging.warning(content_type)
30     if (content_type in list(tipe.keys())):
31         namafile = os.path.basename(url)
32         ekstensi = tipe[content_type]
33         if (tuliskefile):
34             fp = open(f'{tuliskefile}.{ekstensi}', "wb")
35             fp.write(ff.content)
36             fp.close()
37             waktu_process = datetime.datetime.now() - waktu_awal
38             waktu_akhir = datetime.datetime.now()
39             logging.warning(f'writing {tuliskefile}.{ekstensi} dalam waktu {waktu_process} {waktu_awal} s/d {waktu_akhir}')
40             return waktu_process
41     else:
42         return False
43
44  def kirim_gambar(IP_ADDRESS, PORT, filename):
45     print(IP_ADDRESS, PORT, filename)
46     ukuran = os.stat(filename).st_size
47     clientSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
48
49     fp = open(filename, 'rb')
50     k = fp.read()
51     ter kirim = 0
52     for x in k:
53         k_bytes = bytes([x])
54         clientSock.sendto(k_bytes, (IP_ADDRESS, PORT))
55         ter kirim = ter kirim + 1
56
57  if __name__ == '__main__':
58     # check fungsi
59     k = download_gambar('https://m.media-amazon.com/images/M/MV5BYTYxODRmNzI1MGFhNS00MzI4LTk0NTM1YjdiYjgwYmNhODQ2XkEyXkFqcGdeQXVyMTg1ME')
60     print(k)

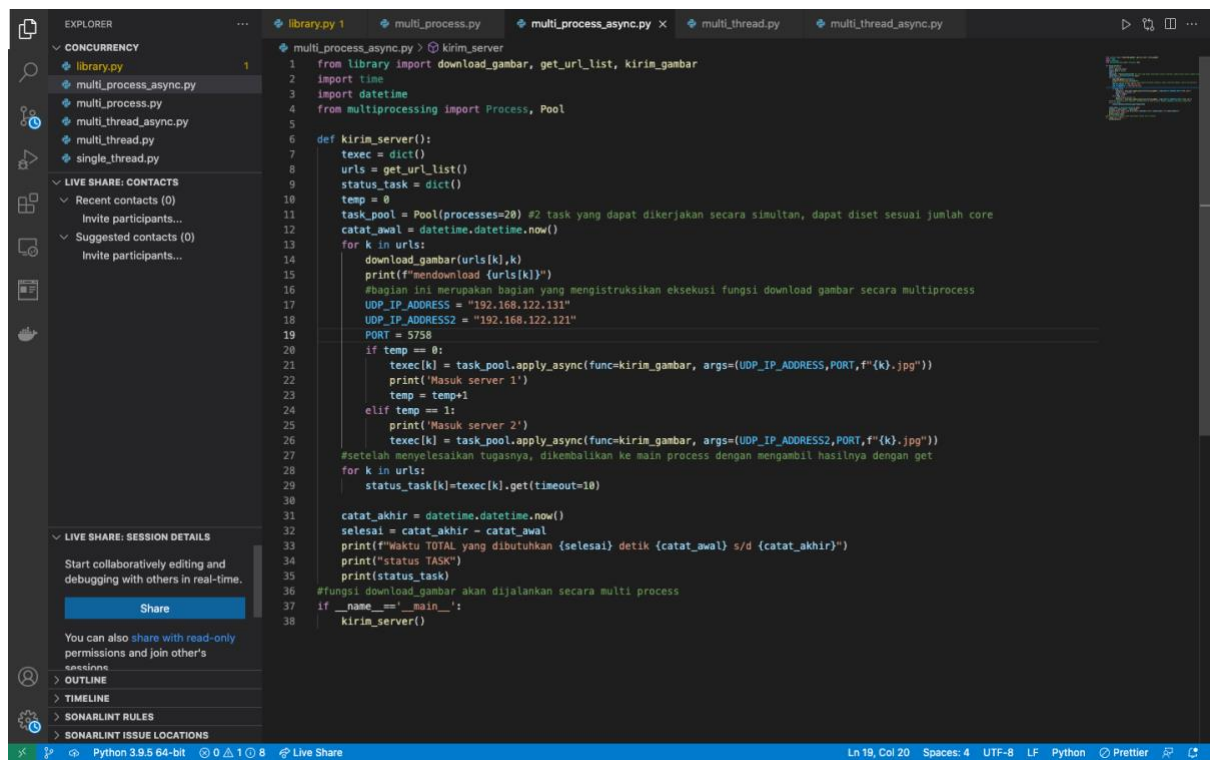
```

```

16  if url is None:
17     return False
18  ff = requests.get(url)
19  tipe = dict()
20  tipe['image/png'] = 'png'
21  tipe['image/jpg'] = 'jpg'
22  tipe['image/gif'] = 'gif'
23  tipe['image/jpeg'] = 'jpg'
24  tipe['application/zip'] = 'zip'
25  tipe['video/quicktime'] = 'mov'
26  # time.sleep(2) # untuk simulasi, diberi tambahan delay 2 detik
27
28  content_type = ff.headers['Content-Type']
29  logging.warning(content_type)
30  if (content_type in list(tipe.keys())):
31      namafile = os.path.basename(url)
32      ekstensi = tipe[content_type]
33      if (tuliskefile):
34          fp = open(f'{tuliskefile}.{ekstensi}', "wb")
35          fp.write(ff.content)
36          fp.close()
37          waktu_process = datetime.datetime.now() - waktu_awal
38          waktu_akhir = datetime.datetime.now()
39          logging.warning(f'writing {tuliskefile}.{ekstensi} dalam waktu {waktu_process} {waktu_awal} s/d {waktu_akhir}')
40          return waktu_process
41  else:
42      return False
43
44  def kirim_gambar(IP_ADDRESS, PORT, filename):
45     print(IP_ADDRESS, PORT, filename)
46     ukuran = os.stat(filename).st_size
47     clientSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
48
49     fp = open(filename, 'rb')
50     k = fp.read()
51     ter kirim = 0
52     for x in k:
53         k_bytes = bytes([x])
54         clientSock.sendto(k_bytes, (IP_ADDRESS, PORT))
55         ter kirim = ter kirim + 1
56
57  if __name__ == '__main__':
58     # check fungsi
59     k = download_gambar('https://m.media-amazon.com/images/M/MV5BYTYxODRmNzI1MGFhNS00MzI4LTk0NTM1YjdiYjgwYmNhODQ2XkEyXkFqcGdeQXVyMTg1ME')
60     print(k)

```

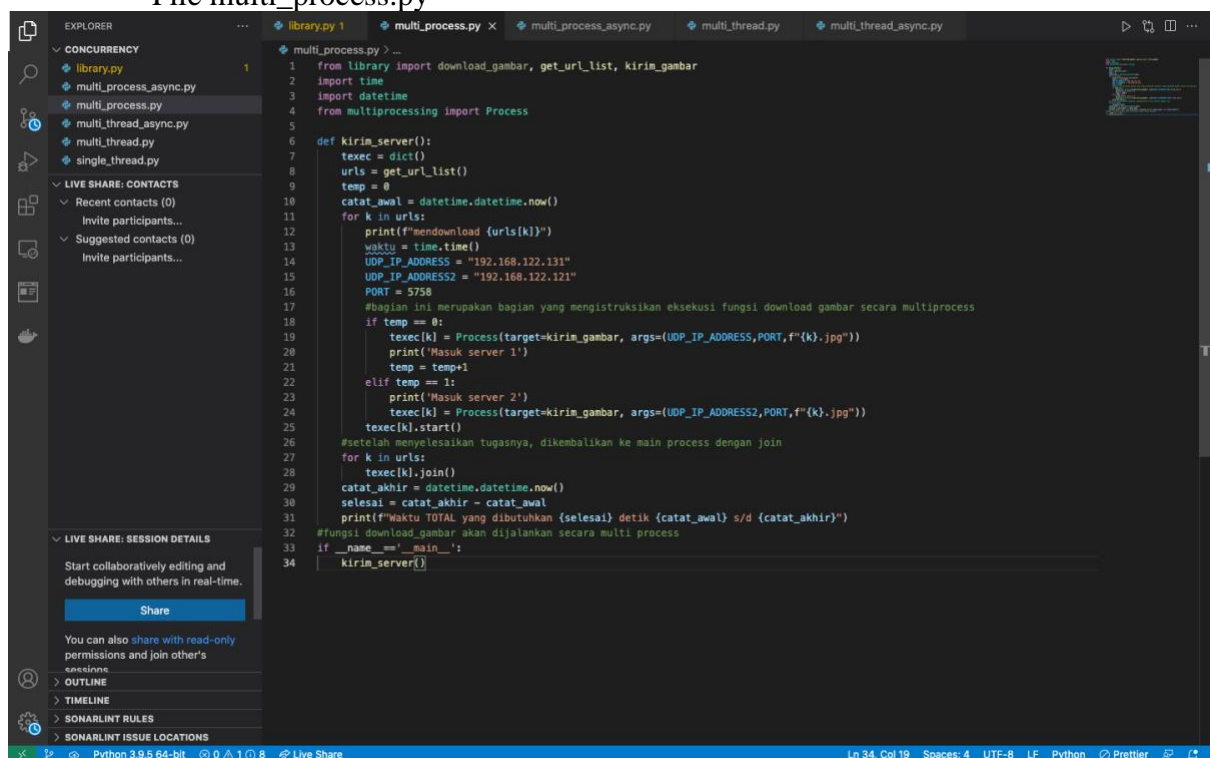
- File multi_process_async.py



The screenshot shows the VS Code editor with the file `multi_process_async.py` open. The Explorer sidebar on the left shows a project structure with a `CONCURRENCY` folder containing `library.py`, `multi_process_async.py`, `multi_process.py`, `multi_thread_async.py`, `multi_thread.py`, and `single_thread.py`. Below this is a `LIVE SHARE: CONTACTS` section and a `LIVE SHARE: SESSION DETAILS` section with a `Share` button. The main editor area displays the code for `multi_process_async.py`, which uses a `Pool` from `multiprocessing` to execute `download_gambar` tasks in parallel. The code includes imports for `library`, `time`, `datetime`, and `multiprocessing`. It defines a `kirin_server` function that sets up a `task_pool` and processes a list of URLs. Comments in Indonesian explain the use of `Pool` for parallel execution and the use of `get` to retrieve results. The code ends with a `if __name__ == '__main__':` block that calls `kirin_server()`.

```
1 from library import download_gambar, get_url_list, kirin_gambar
2 import time
3 import datetime
4 from multiprocessing import Process, Pool
5
6 def kirin_server():
7     texec = dict()
8     urls = get_url_list()
9     status_task = dict()
10    temp = 0
11    task_pool = Pool(processes=20) #2 task yang dapat dikerjakan secara simultan, dapat diset sesuai jumlah core
12    catat_awal = datetime.datetime.now()
13    for k in urls:
14        download_gambar(urls[k],k)
15        print(f"mendownload {urls[k]}")
16        #bagian ini merupakan bagian yang menginstruksikan eksekusi fungsi download gambar secara multiprocessing
17        UDP_IP_ADDRESS = "192.168.122.131"
18        UDP_IP_ADDRESS2 = "192.168.122.121"
19        PORT = 5758
20        if temp == 0:
21            texec[k] = task_pool.apply_async(func=kirin_gambar, args=(UDP_IP_ADDRESS,PORT,f'{k}.jpg'))
22            print('Masuk server 1')
23            temp = temp+1
24        elif temp == 1:
25            print('Masuk server 2')
26            texec[k] = task_pool.apply_async(func=kirin_gambar, args=(UDP_IP_ADDRESS2,PORT,f'{k}.jpg'))
27    #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan mengambil hasilnya dengan get
28    for k in urls:
29        status_task[k]=texec[k].get(timeout=10)
30
31    catat_akhir = datetime.datetime.now()
32    selesai = catat_akhir - catat_awal
33    print(f"Waktu TOTAL yang dibutuhkan (selesai) detik {catat_awal} s/d {catat_akhir}")
34    print("status TASK")
35    print(status_task)
36    #fungsi download_gambar akan dijalankan secara multi process
37    if __name__ == '__main__':
38        kirin_server()
```

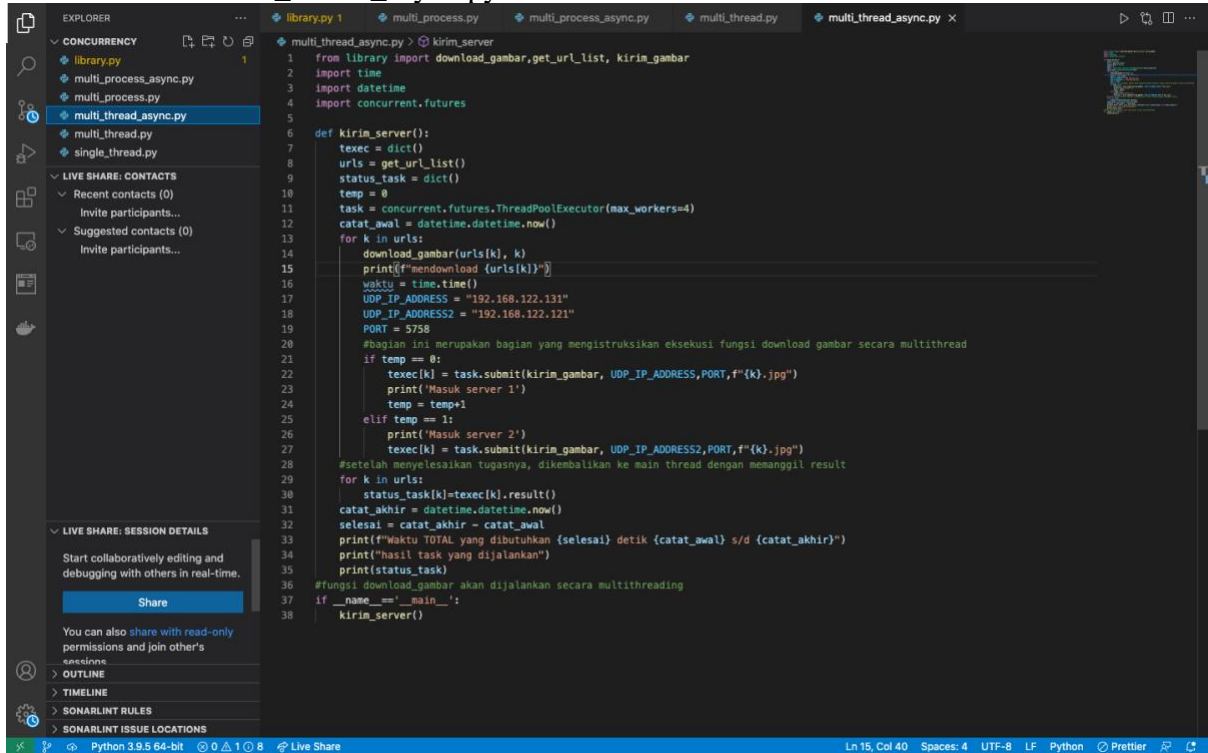
- File multi_process.py



The screenshot shows the VS Code editor with the file `multi_process.py` open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the code for `multi_process.py`, which uses `Process` objects from `multiprocessing` to execute `download_gambar` tasks in parallel. The code includes imports for `library`, `time`, `datetime`, and `multiprocessing`. It defines a `kirin_server` function that sets up a list of `Process` objects and starts them. Comments in Indonesian explain the use of `Process` for parallel execution and the use of `join` to wait for all processes to complete. The code ends with a `if __name__ == '__main__':` block that calls `kirin_server()`.

```
1 from library import download_gambar, get_url_list, kirin_gambar
2 import time
3 import datetime
4 from multiprocessing import Process
5
6 def kirin_server():
7     texec = dict()
8     urls = get_url_list()
9     temp = 0
10    catat_awal = datetime.datetime.now()
11    for k in urls:
12        print(f"mendownload {urls[k]}")
13        waktu = time.time()
14        UDP_IP_ADDRESS = "192.168.122.131"
15        UDP_IP_ADDRESS2 = "192.168.122.121"
16        PORT = 5758
17        #bagian ini merupakan bagian yang menginstruksikan eksekusi fungsi download gambar secara multiprocessing
18        if temp == 0:
19            texec[k] = Process(target=kirin_gambar, args=(UDP_IP_ADDRESS,PORT,f'{k}.jpg'))
20            print('Masuk server 1')
21            temp = temp+1
22        elif temp == 1:
23            print('Masuk server 2')
24            texec[k] = Process(target=kirin_gambar, args=(UDP_IP_ADDRESS2,PORT,f'{k}.jpg'))
25        texec[k].start()
26    #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan join
27    for k in urls:
28        texec[k].join()
29    catat_akhir = datetime.datetime.now()
30    selesai = catat_akhir - catat_awal
31    print(f"Waktu TOTAL yang dibutuhkan (selesai) detik {catat_awal} s/d {catat_akhir}")
32    #fungsi download_gambar akan dijalankan secara multi process
33    if __name__ == '__main__':
34        kirin_server()
```

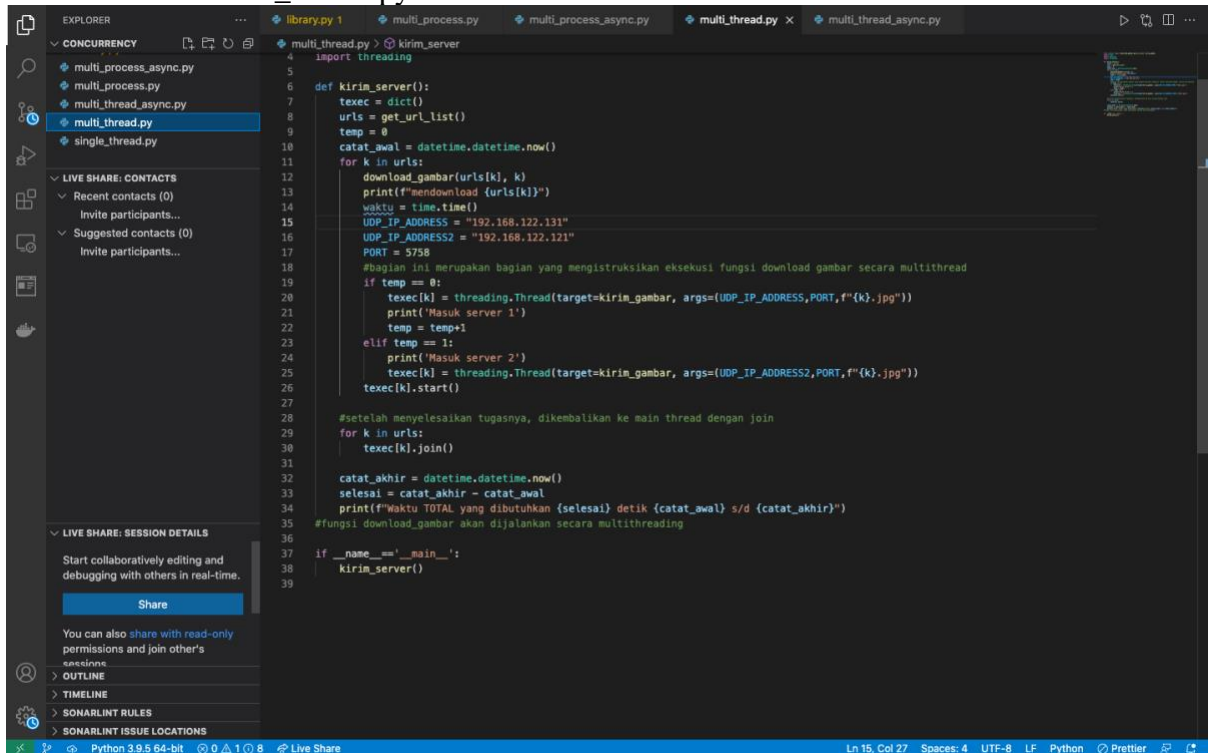

- File multi_thread_async.py



This screenshot shows the Visual Studio Code editor with the file `multi_thread_async.py` open. The Explorer sidebar on the left shows a project structure with files like `library.py`, `multi_process.py`, `multi_thread.py`, and `multi_thread_async.py`. The main editor area displays the code for `multi_thread_async.py`, which uses `concurrent.futures.ThreadPoolExecutor` for asynchronous execution. The code includes a `def kirim_server()` function that downloads images from a list of URLs and prints the total time taken. The status bar at the bottom indicates the file is using Python 3.9.5 64-bit.

```
1 from library import download_gambar, get_url_list, kirim_gambar
2 import time
3 import datetime
4 import concurrent.futures
5
6 def kirim_server():
7     texec = dict()
8     urls = get_url_list()
9     status_task = dict()
10    temp = 0
11    task = concurrent.futures.ThreadPoolExecutor(max_workers=4)
12    catat_awal = datetime.datetime.now()
13    for k in urls:
14        download_gambar(urls[k], k)
15        print(f"mendownload {urls[k]}")
16        waktu = time.time()
17        UDP_IP_ADDRESS = "192.168.122.131"
18        UDP_IP_ADDRESS2 = "192.168.122.121"
19        PORT = 5758
20        #bagian ini merupakan bagian yang menginstruksikan eksekusi fungsi download gambar secara multithread
21        if temp == 0:
22            texec[k] = task.submit(kirim_gambar, UDP_IP_ADDRESS, PORT, f"{k}.jpg")
23            print('Masuk server 1')
24            temp = temp+1
25        elif temp == 1:
26            print('Masuk server 2')
27            texec[k] = task.submit(kirim_gambar, UDP_IP_ADDRESS2, PORT, f"{k}.jpg")
28        #setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan memanggil result
29        for k in urls:
30            status_task[k]=texec[k].result()
31        catat_akhir = datetime.datetime.now()
32        selesai = catat_akhir - catat_awal
33        print(f"Waktu TOTAL yang dibutuhkan (selesai) detik {catat_awal} s/d {catat_akhir}")
34        print("hasil task yang dijalankan")
35        print(status_task)
36        #fungsi download_gambar akan dijalankan secara multithreading
37    if __name__ == '__main__':
38        kirim_server()
```

- File multi_thread.py



This screenshot shows the Visual Studio Code editor with the file `multi_thread.py` open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the code for `multi_thread.py`, which uses `threading.Thread` for synchronous execution. The code includes a `def kirim_server()` function that downloads images from a list of URLs and prints the total time taken. The status bar at the bottom indicates the file is using Python 3.9.5 64-bit.

```
1 import threading
2
3 def kirim_server():
4     texec = dict()
5     urls = get_url_list()
6     temp = 0
7     catat_awal = datetime.datetime.now()
8     for k in urls:
9         download_gambar(urls[k], k)
10        print(f"mendownload {urls[k]}")
11        waktu = time.time()
12        UDP_IP_ADDRESS = "192.168.122.131"
13        UDP_IP_ADDRESS2 = "192.168.122.121"
14        PORT = 5758
15        #bagian ini merupakan bagian yang menginstruksikan eksekusi fungsi download gambar secara multithread
16        if temp == 0:
17            texec[k] = threading.Thread(target=kirim_gambar, args=(UDP_IP_ADDRESS, PORT, f"{k}.jpg"))
18            print('Masuk server 1')
19            temp = temp+1
20        elif temp == 1:
21            print('Masuk server 2')
22            texec[k] = threading.Thread(target=kirim_gambar, args=(UDP_IP_ADDRESS2, PORT, f"{k}.jpg"))
23            texec[k].start()
24
25        #setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan join
26        for k in urls:
27            texec[k].join()
28
29        catat_akhir = datetime.datetime.now()
30        selesai = catat_akhir - catat_awal
31        print(f"Waktu TOTAL yang dibutuhkan (selesai) detik {catat_awal} s/d {catat_akhir}")
32        #fungsi download_gambar akan dijalankan secara multithreading
33    if __name__ == '__main__':
34        kirim_server()
```

- Kemudian jalankan file_server1.py pada alpine-2.

```

< role      alpine-3  X      alpine-2  X      alj >  -  X
concurrency      threading examples
~/Pemrograman_Jaringan_D/progjar3 # cd Jawaban3
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano file_server1.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano file_server1.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # cd
~ # git clone https://github.com/natihdevayanti/Pemrograman_Jaringan_D.git
Cloning into 'Pemrograman_Jaringan_D'...
remote: Enumerating objects: 378, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 378 (delta 19), reused 17 (delta 3), pack-reused 316
Receiving objects: 100% (378/378), 2.20 MiB | 909.00 KiB/s, done.
Resolving deltas: 100% (164/164), done.
~ # cd Pemrograman_Jaringan_D/progjar3
~/Pemrograman_Jaringan_D/progjar3 # ls
Jawaban3      concurrency_asyncio
concurrency      threading examples
~/Pemrograman_Jaringan_D/progjar3 # cd Jawaban3
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # ls
file_server1.py      multi_process.py      multi_thread_async.py
file_server2.py      multi_process_async.py  single_thread.py
library.py           multi_thread.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 file_server1.py

```

- Jalankan file_server2.py pada alpine-3.

```

< role      alpine-3  X      alpine-2  X      alj >  -  X
Jawaban3      concurrency_asyncio
concurrency      threading examples
~/Pemrograman_Jaringan_D/progjar3 # cd Jawaban3
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano file_server2.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano file_server2.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # cd
~ # git clone https://github.com/natihdevayanti/Pemrograman_Jaringan_D.git
Cloning into 'Pemrograman_Jaringan_D'...
remote: Enumerating objects: 378, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 378 (delta 19), reused 17 (delta 3), pack-reused 316
Receiving objects: 100% (378/378), 2.20 MiB | 343.00 KiB/s, done.
Resolving deltas: 100% (172/172), done.
~ # cd Pemrograman_Jaringan_D/progjar3
~/Pemrograman_Jaringan_D/progjar3 # cd Jawaban3
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # ls
file_server1.py      multi_process.py      multi_thread_async.py
file_server2.py      multi_process_async.py  single_thread.py
library.py           multi_thread.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 file_server3.py
python3: can't open file 'file_server3.py': [Errno 2] No such file or directory
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 file_server2.py

```

- Lalu install module requests pada alpine-1

```
< alpine-3 x alpine-2 x alpine-1 x > - x
python3: can't open file 'pip': [Errno 2] No such file or directory
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 -m pip install requests
Collecting requests
  Downloading requests-2.26.0-py2.py3-none-any.whl (62 kB)
    |████████████████████████████████████████| 62 kB 72 kB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2021.5.30-py2.py3-none-any.whl (145 kB)
    |████████████████████████████████████████| 145 kB 458 kB/s
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
    |████████████████████████████████████████| 138 kB 667 kB/s
Collecting idna<4,>=2.5
  Downloading idna-3.2-py3-none-any.whl (59 kB)
    |████████████████████████████████████████| 59 kB 635 kB/s
Collecting charset-normalizer~=2.0.0
  Downloading charset_normalizer-2.0.1-py3-none-any.whl (35 kB)
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2021.5.30 charset-normalizer-2.0.1 idna-3.2 requests-2.26.0 urllib3-1.26.6
WARNING: You are using pip version 21.0.1; however, version 21.1.3 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 #
```

- Jalankan file multi_process_async.py pada alpine-1.

```
< alpine-3 x alpine-2 x alpine-1 x > - x
KeyboardInterrupt
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano multi_process_async.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano library.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 multi_process_async.py
WARNING:root:image/jpeg
WARNING:root:writing olivia.jpg dalam waktu 0:00:00.466766 2021-07-14 14:33:15.274458 s/d 2021-07-14 14:33:15.741243
mendownload https://m.media-amazon.com/images/M/MV5BYTYxODRmNzItMGFkNS00Mzk4LTk0NTMtYjdiYjgwYWNhODQ2XkEyXkFqcGdeQXVyNTg1MDY4NjQ0._V1_UY1200_CR285,0,630,1200_AL_.jpg
192.168.122.131 5758 olivia.jpg
Masuk server 1
WARNING:root:image/jpeg
WARNING:root:writing taylor.jpg dalam waktu 0:00:03.886661 2021-07-14 14:33:15.756583 s/d 2021-07-14 14:33:19.643256
mendownload http://jadiberita.com/wp-content/uploads/2014/06/taylor-swift-presenting-jpg.jpg
Masuk server 2
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:07.758355 detik 2021-07-14 14:33:15.274350 s/d 2021-07-14 14:33:23.032705
status TASK
{'olivia': None, 'taylor': None}
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 #
```


- Jalankan file multi_process.py pada alpine-1.

```

< alpine-3 x alpine-2 x alpine-1 x > - x
WARNING:root:image/jpeg
WARNING:root:writing taylor.jpg dalam waktu 0:00:03.886661 2021-07-14 14:33:15.7
56583 s/d 2021-07-14 14:33:19.643256
mendownload http://jadiberita.com/wp-content/uploads/2014/06/taylor-swift-presen
ting-jpg.jpg
Masuk server 2
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:07.758355 detik 2021-07-14 14:33:15.274350 s/d
2021-07-14 14:33:23.032705
status TASK
{'olivia': None, 'taylor': None}
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 multi_process.py
mendownload https://m.media-amazon.com/images/M/MV5BYTYxODRmNzItMGFkNS00Mzk4LTk0
NTMtYjdiYjgwYWNhODQ2XkEyXkFqcGdeQXVyNTg1MDY4NjQ@._v1_UY1200_CR285,0,630,1200_AL
.jpg
Masuk server 1
mendownload http://jadiberita.com/wp-content/uploads/2014/06/taylor-swift-presen
ting-jpg.jpg
Masuk server 2
192.168.122.131 5758 olivia.jpg
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:07.053062 detik 2021-07-14 14:34:07.598888 s/d
2021-07-14 14:34:14.651950
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 #

```

- Jalankan file multi_thread_async.py pada alpine-1.

```

< alpine-3 x alpine-2 x alpine-1 x > - x
ction object at 0x7fbce1beaa30>: Failed to establish a new connection: [Errno -3
] Try again'))
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # nano multi_thread_async.py
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 multi_thread_async.py
WARNING:root:image/jpeg
WARNING:root:writing olivia.jpg dalam waktu 0:00:00.353731 2021-07-14 14:37:44.2
10474 s/d 2021-07-14 14:37:44.564272
mendownload https://m.media-amazon.com/images/M/MV5BYTYxODRmNzItMGFkNS00Mzk4LTk0
NTMtYjdiYjgwYWNhODQ2XkEyXkFqcGdeQXVyNTg1MDY4NjQ@._v1_UY1200_CR285,0,630,1200_AL
.jpg
192.168.122.131 5758 olivia.jpg
Masuk server 1
WARNING:root:image/jpeg
WARNING:root:writing taylor.jpg dalam waktu 0:00:07.654172 2021-07-14 14:37:44.5
66287 s/d 2021-07-14 14:37:52.220472
mendownload http://jadiberita.com/wp-content/uploads/2014/06/taylor-swift-presen
ting-jpg.jpg
Masuk server 2
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:11.600680 detik 2021-07-14 14:37:44.210459 s/d
2021-07-14 14:37:55.811139
hasil task yang dijalankan
{'olivia': None, 'taylor': None}
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # █

```

- Jalankan file multi_thread.py pada alpine-1.

```

< alpine-3 x alpine-2 x alpine-1 x > - x
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:11.600680 detik 2021-07-14 14:37:44.210459 s/d
2021-07-14 14:37:55.811139
hasil task yang dijalankan
{'olivia': None, 'taylor': None}
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # python3 multi_thread.py
WARNING:root:image/jpeg
WARNING:root:writing olivia.jpg dalam waktu 0:00:00.250316 2021-07-14 14:38:29.7
01878 s/d 2021-07-14 14:38:29.952211
mendownload https://m.media-amazon.com/images/M/MV5BYTYxODRmNzItMGFkNS00Mzk4LTk0
NTMtYjdiYjgwYWNhODQ2XkEyXkFqcGdeQXVyNTg1MDY4NjQ0._V1_UY1200_CR285,0,630,1200_AL
.jpg
Masuk server 1
192.168.122.131 5758 olivia.jpg
WARNING:root:image/jpeg
WARNING:root:writing taylor.jpg dalam waktu 0:00:07.622443 2021-07-14 14:38:29.9
60511 s/d 2021-07-14 14:38:37.582972
mendownload http://jadiberita.com/wp-content/uploads/2014/06/taylor-swift-presen
ting-jpg.jpg
Masuk server 2
192.168.122.121 5758 taylor.jpg
Waktu TOTAL yang dibutuhkan 0:00:11.337049 detik 2021-07-14 14:38:29.701865 s/d
2021-07-14 14:38:41.038914
~/Pemrograman_Jaringan_D/progjar3/Jawaban3 # █

```

- Maka, pada alpine-2 dan alpine-3 sebagai Server akan muncul hasil sebagai berikut.

```

< .ole alpine-3 x alpine-2 x alpine-1 x > - x
('192.168.122.118', 36162) 19076 1 b'\xe6'
('192.168.122.118', 36162) 19077 1 b'\xe7'
('192.168.122.118', 36162) 19078 1 b'\r'
('192.168.122.118', 36162) 19079 1 b'\xb8'
('192.168.122.118', 36162) 19080 1 b'_'
('192.168.122.118', 36162) 19081 1 b'\xa4'
('192.168.122.118', 36162) 19082 1 b'\x98'
('192.168.122.118', 36162) 19083 1 b'w'
('192.168.122.118', 36162) 19084 1 b'\xce'
('192.168.122.118', 36162) 19085 1 b'\xce'
('192.168.122.118', 36162) 19086 1 b'\x83'
('192.168.122.118', 36162) 19087 1 b'q'
('192.168.122.118', 36162) 19088 1 b'\x04'
('192.168.122.118', 36162) 19089 1 b'\xb5'
('192.168.122.118', 36162) 19090 1 b'\xb3'
('192.168.122.118', 36162) 19091 1 b'\xe1'
('192.168.122.118', 36162) 19092 1 b'\x00'
('192.168.122.118', 36162) 19093 1 b'Q'
('192.168.122.118', 36162) 19094 1 b'\xcb'
('192.168.122.118', 36162) 19095 1 b'\xb2'
('192.168.122.118', 36162) 19096 1 b'w'
('192.168.122.118', 36162) 19097 1 b'\xcf'
('192.168.122.118', 36162) 19098 1 b'p'
('192.168.122.118', 36162) 19099 1 b'b'

```

```
< role alpine-3 x alpine-2 x alj > - x
('192.168.122.118', 41207) 11260 1 b'\x15'
('192.168.122.118', 41207) 11261 1 b'\x8b'
('192.168.122.118', 41207) 11262 1 b'\xf8'
('192.168.122.118', 41207) 11263 1 b'\xbf'
('192.168.122.118', 41207) 11264 1 b'\x91'
('192.168.122.118', 41207) 11265 1 b'\xf3'
('192.168.122.118', 41207) 11266 1 b'\xff'
('192.168.122.118', 41207) 11267 1 b'\x00'
('192.168.122.118', 41207) 11268 1 b'\xea'
('192.168.122.118', 41207) 11269 1 b'"'"'
('192.168.122.118', 41207) 11270 1 b'\xb8'
('192.168.122.118', 41207) 11271 1 b'\xb2'
('192.168.122.118', 41207) 11272 1 b'\xfe'
('192.168.122.118', 41207) 11273 1 b'\xa3'
('192.168.122.118', 41207) 11274 1 b'>'
('192.168.122.118', 41207) 11275 1 b'\xa7'
('192.168.122.118', 41207) 11276 1 b'\x07'
('192.168.122.118', 41207) 11277 1 b'n'
('192.168.122.118', 41207) 11278 1 b'\xd4'
('192.168.122.118', 41207) 11279 1 b'\xe1'
('192.168.122.118', 41207) 11280 1 b'\x7f'
('192.168.122.118', 41207) 11281 1 b'\x8e'
('192.168.122.118', 41207) 11282 1 b'\xe2'
('192.168.122.118', 41207) 11283 1 b'\xa5'
```