به نام خدا

گزارش آزمایش 4 آزمایشگاه سیستم عامل

محمد جواد زندیه

بخش اول:

```python
from multiprocessing import shared_memory
import array

# create shared memory with size 20
shm_a = shared_memory.SharedMemory(create=True, size=20)

# put something in buffer of shared memory
buffer = shm_a.buf
buffer[:] = bytearray([i for i in range(20)])

# attach to an existing shared memory block
shm_b = shared_memory.SharedMemory(shm_a.name)

# copy data of buffer into an array and print it
data = array.array('d', shm_b.buf[:])
print(data[2:8])

# close access to shared memory from instances b and a
shm_b.close()
shm_a.close()

# destroy shared memory block
shm_a.unlink()
```

در این بخش از ما خواسته شده است که یک shared memory برای دو پردازه در نظر بگیریم و توسط یکی از آنها عمل نوشتن و توسط دیگری عمل خوندن از این حافظه مشترک صورت بگیرد. Shm_a حافظه مشترکی است که توسط پردازه اول ساخته شده و توی بافر آن اطلاعاتی را قرار دادیم. Shm_b دسترسی به حافظه مشترک را امکان پذیر می کنید و می توان اطلاعات داخل حافظه مشترک را از داخل پردازه دوم خواند و چاپ کرد. در انتها هم دسترسی دو پردازه به حافظه مشترک را قطع کرده و در نهایت حافظه تخصیص داده شده برای داده های مشترک را unallocate یا destroy می کنیم.

از داخل پردازه دوم مقدار موجود در بافر از ایندکس 2 تا 8 را خوانده و چاپ کرده ایم که به صورت زیر نشان داده شده است:

```
javad@javad-HP-350-G1:~/Desktop/OSLab/project4$ /bin/python3 /home/javad/Desktop/OSLab/project4/part1.py
array('d', [2.0, 3.0, 4.0, 5.0, 6.0, 7.0])
javad@javad-HP-350-G1:~/Desktop/OSLab/project4$
```

در بخش دوم از ما خواسته شده که یک chat application بسازیم که یک سرور داشته باشم و چندین کلاینت بتوانند به آن وصل شوند. هر کاربر باید ابتدا نام خود را به سرور اعلام کند و سپس می تواند از میان عملیات های join, send, leave, quit استفاده کند.

(مواردی که داخل دستور کار خواسته شده بود پیاده سازی شده اند، همچنین در سمت سرور برای آنکه دید بهتری نسبت به فرایند داشته باشیم تعدادی log در هر مرحله انداخته می شود که در ادامه توضیح خواهم داد)

بخش سرور:

part2_server.py > client_handler

```python
"""Server for multithreaded (asynchronous) chat application."""
from socket import AF_INET, socket, SOCK_STREAM
from _thread import *

# handling a single client connection
def client_handler(client):  # Takes client socket as argument.
    """client commands:
        join [groupId]
        send [groupId] [message]
        leave [groupId]
        quit
    """
    try:
        name = client.recv(BUFSIZ).decode("utf8")
        print(name, ": login to app")
        clinets[name] = client
        while True:
            command = client.recv(BUFSIZ).decode("utf8")
            command = command.split()

            if command[0] == "quit":
                print(name, ": quit from app")
                client.close()
                break

            elif command[0] == "join":
                if name not in groups[command[1]]:
                    groups[command[1]].append(name)
```

part2_server.py > client_handler

```
29
30                  elif command[0] == "send":
31                      if name in groups[command[1]]:
32                          for c_name in groups[command[1]]:
33                              message = str(name + ": " + ' '.join(command[2:]))
34                              clinets[c_name].send(bytes(message, "utf8"))
35
36                  elif command[0] == "leave":
37                      if name in groups[command[1]]:
38                          groups[command[1]].remove(name)
39
40                  print(groups)
41          finally:
42              client.close()
43
44
45  groups = {"1":[], "2":[], "3":[]} # some groups of users
46  clinets = {}
47  BUFSIZ = 1024 # buffer size
48
49  """create a socket
50      AF_INET : address domain of the socket
51      SOCK_STREAM : means that data or characters are read in a continuous flow"""
52  SERVER = socket(AF_INET, SOCK_STREAM)
53
54  # try to bind an address
55  try:
56      # host = '127.0.0.1, port = 1234, address = (host, port)
57      SERVER.bind(('127.0.0.1', 1234))
58  except socket.error as e:
59      print(str(e))
```

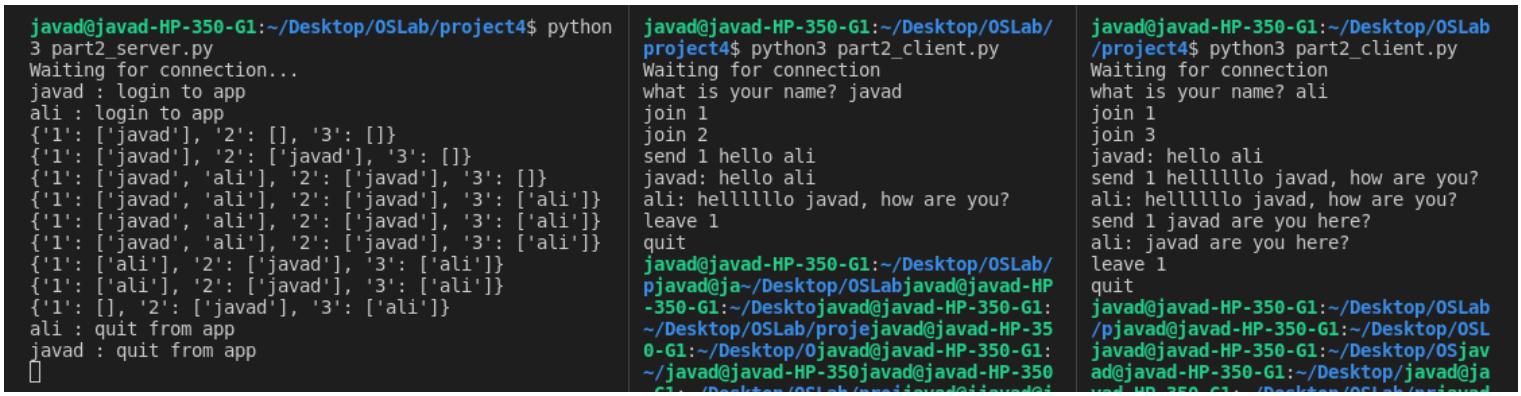
```
60
61  # listen for clinet request
62  print("Waiting for connection...")
63  SERVER.listen()
64
65  try:
66      while True:
67          client, client_address = SERVER.accept()
68          start_new_thread(client_handler, (client, ))
69  finally:
70      SERVER.close()
```

بخش کلاینت:

```python
from socket import AF_INET, socket, SOCK_STREAM
from _thread import *

def read_server_message(CLIENT):
    while True:
        Response = CLIENT.recv(BUFSIZ)
        if not Response:
            pass
        else:
            print(Response.decode('utf-8'))



CLIENT = socket(AF_INET, SOCK_STREAM)
print('Waiting for connection')

try:
    CLIENT.connect(('127.0.0.1', 1234))
except socket.error as e:
    print(str(e))

BUFSIZ = 1024 # buffer size

# send name to server
Input = input("what is your name? ")
CLIENT.send(str.encode(Input))

```

```python
 try:
    # check for response
    start_new_thread(read_server_message, (CLIENT, ))
    while True:
        # send command
        Input = input()
        CLIENT.send(str.encode(Input))
        if Input == "quit":
            break
 finally:
    CLIENT.close()
```

خروجی از یک نمونه کار با chat application :



طبق لاگ هایی که در بخش سرور وجود دارد میتوان دید که ابتدا کاربری با نام javad

(ترمینال وسط) وارد ارتباط با سرور می شود. سپس هم ali (ترمینال راست) وارد سامانه می شود. Javad در گروه های 1 و 2 عضو می شود. سپس ali در گروه های 1 و 3 عضو می شود.

Javad در گروه 1 پیام hello ali را می نویسد و چون javad , ali هر دو در گروه 1 هستند پیام برایشان نشان داده می شود. ali هم در جواب پیامی در گروه 1 می فرستد که آن هم برای هر دو نشان داده می شود چون هر دو در گروه 1 عضو هستند. بعد از این javad گروه 1 را ترک می کند و سپس که ali برایش در گروه 1 پیام میفرستد نمیتواند آنرا ببیند چون آنرا ترک کرده است. پس ali هم گروه 1 را ترک می کند و از سامانه خارج می شود و javad هم از سامانه خارج می شود اما همچنان سرور پابرجاست که اگر بعدا کاربری خواست وارد شود بتواند به او پاسخ بدهد.