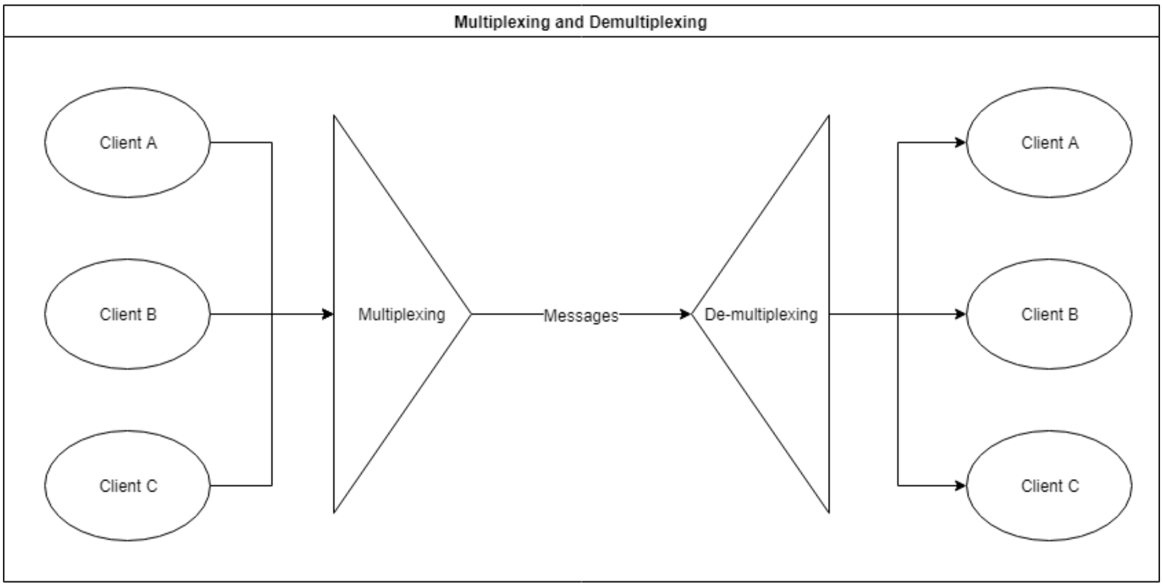
Multiplexing in Python3

Date: 2020-11-03 00:11:20

What is Multiplexing and De-multiplexing?



Multiplexing is generally representing a concept of transferring multiple messages by the same road. By merging the slow road to a faster road, can efficiently bring advantage to the usage of the faster road. After the data has proceeded when the messages arrive at the destination server, we need to use De-multiplexing to the different sources. The advantage of multiplexing that includes saving the cost and efficiently use the resources for communication.

Server-side example

```
import socket
import queue
import select

buffer = 1024
available_connections = 1000
server = socket.socket()
server.bind(('localhost', 8888))
server.listen(available_connections)
server.setblocking(False)
message_redirect = {}
inputs = [server, ]
outputs = []


def add_readable(readable):
    global buffer
    global inputs
    global outputs
    global server
    global message_redirect
    for r in readable:
        if r is server:
            conn, addr = server.accept()
            inputs.append(conn)
            message_redirect[conn] = queue.Queue()
        else:
            data = r.recv(buffer)
            message_redirect[r].put(data)
            outputs.append(r)


def add_writable(writeable):
    global outputs
    global message_redirect
    for w in writeable:
        data_to_client = message_redirect[w].get()
        w.send(data_to_client)
        outputs.remove(w)
```

```
def exception_handler():
    global inputs
    global outputs
    global message_redirect
    for e in exceptional:
        if e in outputs:
            outputs.remove(e)
        inputs.remove(e)
        del message_redirect[e]


while True:
    readable, writeable, exceptional = select.select(inputs, outputs, inputs)
    add_readable(readable)
    add_writable(writeable)
    exception_handler()
```

Client-side example (Test client)

```
import socket
import time
import socket
import sys

buffer = 1024
host = '127.0.0.1'
port = 8888


def send_msg(sock):
    global buffer
    while True:
        sock.send(bytes(str.encode('test')))
        msg = sock.recv(buffer)
        print(msg.decode('utf-8'))
        time.sleep(2)


def main():
    global host
    global port
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
        sock.connect((host, port))
        send_msg(sock)


if __name__ == "__main__":
    main()
```