

DOKUMENTASI PEMROGRAMAN JARINGAN

## PROTOKOL PADA SISTEM CHATTING



KELAS B

Alvin Lazaro

5113100067

Asisten:

Bahrul Halimi

Dosen:

Royyana Muslim I.

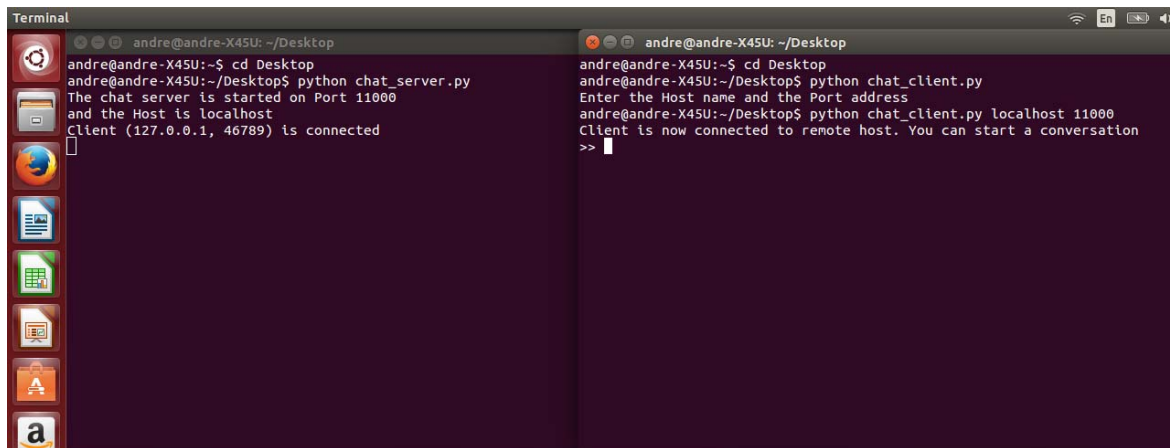
INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

2015

## Latar Belakang

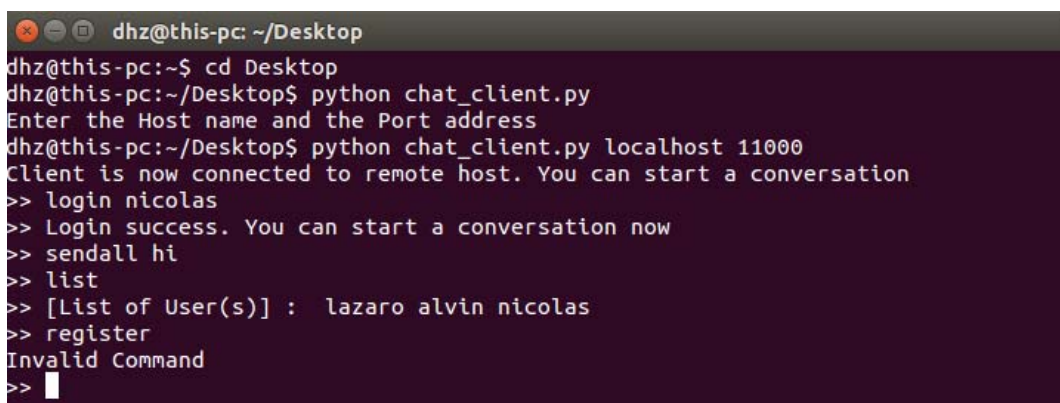
Program chatting ini dibuat dengan menggunakan bahasa pemrograman python. Terdapat 2 source-code yaitu chat\_server.py dan chat\_client.py. File chat\_server.py bertindak sebagai server untuk komunikasi 2 atau lebih client (chat\_client.py). Berikut ini adalah penjelasan tentang program chatting tersebut:

1. File chat\_server.py harus di-compile terlebih dahulu untuk menyediakan Host dan Port bagi client.
2. Kemudian file chat\_client.py di-compile pada 3 terminal yang berbeda. Sehingga seolah-olah terdapat 3 client yang saling berkomunikasi satu sama lain. Username Host dan alamat Port sudah di-set secara permanen pada code chat\_server.py dengan Host adalah localhost dan Port adalah 11000.
3. Untuk melakukan chatting, client harus memasukkan username Host dan alamat Port sesuai dengan yang di-set pada server. Dalam hal ini Host-nya adalah "localhost" dan Port-nya adalah "11000".



Gambar 1. Tampilan pada server (sebelah kiri) dan client (sebelah kanan) saat client sudah memasukkan Host dan Port

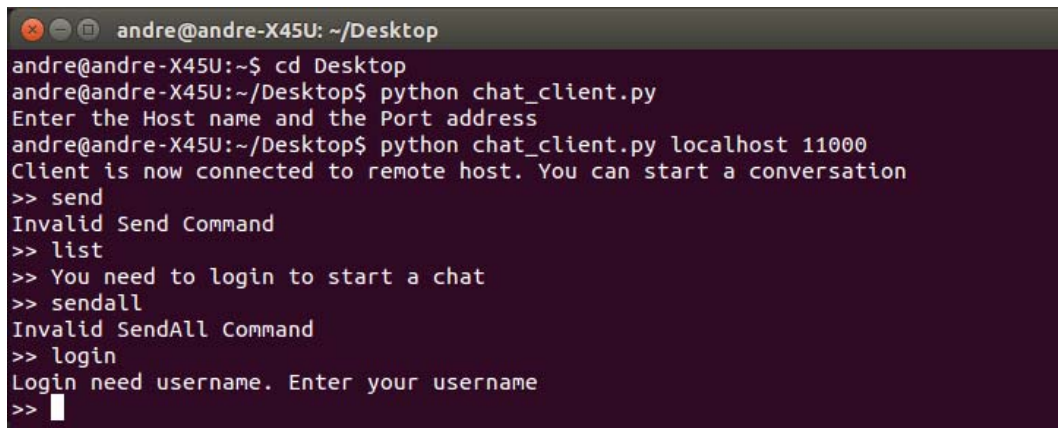
Perlu diketahui bahwa server memiliki 4 buah protokol untuk chatting. Keempat protokol tersebut adalah login, send, sendall, list, dan whoami. Jika client memasukkan protokol lain (misalnya register), server akan mengirimkan pesan "Invalid Command" pada layar client.



Gambar 2. Karena tidak ada protokol "register" ditampilkan pesan "Invalid Command" pada layar client

# Login

Untuk melakukan chatting, client harus melakukan login terlebih dahulu. Tanpa melakukan login, client tidak dapat menjalankan perintah-perintah lainnya. Dalam melakukan login, client wajib menuliskan username. Tanpa username, server akan mengirimkan pesan "Login need username. Enter your username".



```
andre@andre-X45U: ~/Desktop
andre@andre-X45U:~$ cd Desktop
andre@andre-X45U:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
andre@andre-X45U:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> send
Invalid Send Command
>> list
>> You need to login to start a chat
>> sendall
Invalid SendAll Command
>> login
Login need username. Enter your username
>> 
```

Gambar 3. Client harus login terlebih dahulu, tanpa login perintah lain tidak dapat dieksekusi

Berikut ini adalah potongan code dari protokol login:

```
temp1 = string.split(data[:-1])
d=len(temp1)
if temp1[0]=="login" :

    log_in(sock, str(temp1[1]))
```

Syntax untuk login adalah login [spasi] username. Contohnya, "login lazaro" yang berarti client melakukan login dengan username lazaro. Variable temp1 akan dipecah menjadi kata per kata. Sehingga index ke-0 dari temp1 adalah "login" dan index ke-1 adalah "lazaro". Dalam potongan code tersebut, akan dilakukan pengecekan pada index ke-0 variable temp1. Jika terdeteksi index ke-0 dari temp1 adalah "login", dijalankan fungsi log\_in. Proses pemecahan temp1 ini berlaku pada protokol-protokol lainnya dalam system chatting ini. Sehingga, pada bagian awal dari seluruh protokol dalam system chatting ini akan selalu terdapat prosedur ini. Berikut ini adalah isi dari fungsi log\_in:

```
def log_in (sock, user):

    g = 0
    f = 0

    for name in NAME_LIST:

        if name == user:

            g = 1

        if name == sock:

            f = 1
```

```

    if f==1:

        send_msg(sock, "You already have a username\n")

    elif g==1:

        send_msg(sock, "Username already exist. Enter another
name\n")

    else:

        NAME_LIST.append(sock)
        NAME_LIST.append(user)
        send_msg(sock, "Login success. You can start a conversation
now\n")

```

Pada fungsi `log_in`, terdapat variable `g` dan `f`. variable `g` berfungsi untuk menyimpan alamat port client dan variable `g` berfungsi untuk menyimpan username client. Fungsi ini dapat mendeteksi kondisi di mana client melakukan login kembali (sebelumnya sudah melakukan login) pada bagian “if f==1”. Jika kondisi ini terjadi, server akan menampilkan pesan “You already have a username” pada layar client.

Selain itu, fungsi ini juga dapat mendeteksi apabila terdapat client yang akan login dengan menggunakan username yang sama dengan username client lain yang telah terdaftar dalam array `NAME_LIST` pada bagian “if g==1”. Hal ini dilakukan untuk mencegah adanya kesamaan username pada 2 client yang berbeda. Jika kondisi ini terjadi, server akan menampilkan pesan “Username already exist. Enter another name” pada layar client.

Jika kedua kondisi tersebut tidak terjadi, alamat port dan username client akan disimpan dalam sebuah array berusername `NAME_LIST`. Penyimpanan dilakukan dengan urutan alamat, username, alamat, username, dan seterusnya. Urutan alamat port dan username client pada array `NAME_LIST` adalah berdasarkan waktu login sebuah client. Jadi alamat port dan username client terdepan adalah alamat port dan username client dari client yang login pertama kali ke server. Dan server akan menampilkan pesan “Login success. You can start a conversation” pada layar client yang berhasil melakukan login.

```
andre@andre-X45U: ~/Desktop
andre@andre-X45U:~$ cd Desktop
andre@andre-X45U:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
andre@andre-X45U:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> send
Invalid Send Command
>> list
>> You need to login to start a chat
>> sendall
Invalid SendAll Command
>> login
Login need username. Enter your username
>> login lazaro
>> Login success. You can start a conversation now
>> █
```

Gambar 4. Tampilan jika client sukses melakukan login

## Send

Protokol send digunakan untuk mengirimkan pesan ke client yang spesifik. Artinya, protokol ini hanya mengirimkan pesan ke satu client sesuai dengan permintaan client yang menuliskannya. Syntax dari send adalah send [spasi] username client tujuan [spasi] pesan yang disampaikan. Jika salah satu komponen tidak tertulis (misalnya tidak menuliskan username client tujuan), akan ditampilkan pesan "Invalid Send Command" pada layar client.

```
>> send
Invalid Send Command
```

Gambar 5. Tampilan pesan yang dikirimkan server jika client tidak menuliskan username client tujuan dan pesan

Contohnya, "send lazaro hello" yang berarti mengirimkan pesan "hello" kepada client dengan username "lazaro". Berikut ini adalah potongan code untuk send.

```
elif temp1[0]=="send" :
    logged = 0
    user = ""
    for x in range (len(NAME_LIST)):
        if NAME_LIST[x]==sock:
            logged=1
            user=NAME_LIST[x+1]
```

Seperti pada protokol login, temp1 akan dipecah menjadi kata per kata. Dalam kasus ini, kata pada index ke-0 adalah "send", index ke-1 adalah "lazar", dan index ke-2 adalah "hello". Jika index ke-0 berisi kata "send", protokol akan dijalankan dengan urutan sebagai berikut.

Pada awalnya, logged di-set menjadi 0 terlebih dahulu. Variable logged berfungsi sebagai status apakah client sudah login atau belum. Kemudian dilakukan pengecekan sebanyak x (variable iterator sebanyak isi array NAME\_LIST) untuk mengecek apakah client sudah melakukan login atau belum. Jika alamat socket client terdapat pada array NAME\_LIST, client sudah melakukan login dan status logged akan diubah menjadi 1 serta username client akan dimasukkan ke variable user. Tetapi jika tidak ditemukan pada array NAME\_LIST, akan ditampilkan pesan "You need to login to start a chat" pada layar client. Kemudian dilanjutkan dengan perintah berikut.

```
if logged==0:
    send_msg(sock, "You need to login to start a chat\n")
else:
    temp2=""
    for x in range (len(temp1)):
        if x>1:
            if not temp2:
                temp2+=str(temp1[x])
            else:
                temp2+=" "
                temp2+=str(temp1[x])
```

Setelahnya, dilakukan pemrosesan terhadap pesan yang akan dikirimkan oleh client. Pesan yang sebelumnya ditampung pada variable temp1 mulai dari index ke-2 akan dipindahkan ke variable temp2. Hal ini dilakukan pada perulangan for sebanyak x (variable iterator sebanyak isi array NAME\_LIST). Jika temp2 masih kosong, temp2 akan langsung diisi dengan kata pertama pada pesan yang akan dikirimkan. Tetapi jika temp2 sudah terisi, temp2 akan diisi dengan spasi dan kata berikutnya dari rangkaian pesan yang akan dikirimkan. Hal ini dilakukan hingga semua pesan telah dimasukkan ke dalam temp2. Kemudian dilanjutkan dengan perintah berikut.

```
for x in range (len(NAME_LIST)):
    if NAME_LIST[x]==temp1[1]:
        send_msg(NAME_LIST[x-1], "["+user+"] : "+temp2+"\n")
```

Perintah ini digunakan untuk mengirimkan pesan ke client tujuan. Perintah ini akan dijalankan jika username client tujuan yang terdapat pada index ke-1 temp1 telah tercantum pada array NAME\_LIST. Jika tercantum, dijalankan fungsi send\_msg. Nantinya, client tujuan akan menerima pesan yang dikirimkan oleh client pengirim lengkap dengan username client pengirimnya.

**Gambar 6. Contoh hasil dari protokol send. Client "alvin" (sebelah kiri bawah) mengirim pesan "hello" ke client "lazarus"**

Berikut ini adalah isi dari fungsi `send_msg`.

```
def send_msg (sock, message):
    try:
        sock.send(message)
    except:
        sock.close()
        if sock in SOCKET_LIST:
            SOCKET_LIST.remove(sock)
```

## Send All

Protokol send all (dalam chatting ditulis sendall) digunakan untuk mengirimkan pesan ke semua client yang terdaftar. Artinya, protokol ini mengirimkan pesan ke semua client yang tercantum dalam array NAME\_LIST (kecuali sang pengirim sendiri). Syntax dari send all adalah sendall [spasi] pesan yang disampaikan. Jika salah satu komponen tidak tertulis (misalnya tidak menuliskan pesan) atau terdapat kesalahan penulisan (misalkan “sendall” ditulis menjadi “send all”), akan ditampilkan pesan “Invalid SendAll Command” pada layar client.

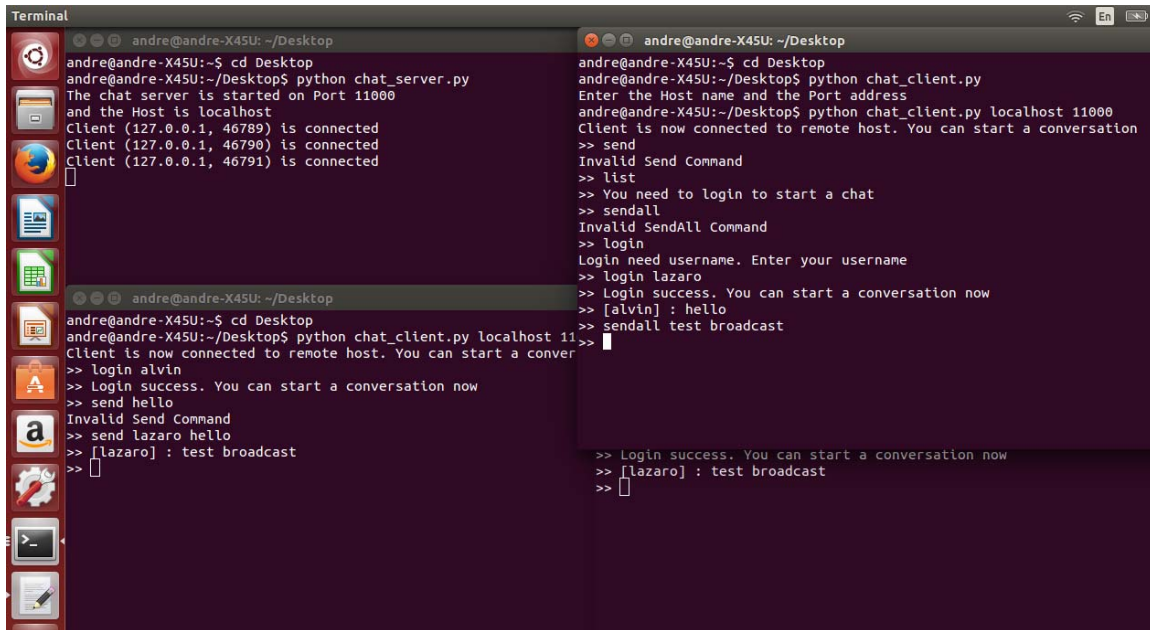
**Gambar 7. Tampilan jika client tidak memasukkan pesan setelah perintah sendall**

Contohnya, “sendall test broadcast” yang berarti mengirimkan pesan “test broadcast” kepada semua client kecuali client pengirim. Berikut ini adalah potongan code untuk sendall.

```
elif temp1[0]=="sendall" :  
    logged = 0  
    user = ""  
  
    for x in range (len(NAME_LIST)):  
        if NAME_LIST[x]==sock:  
            logged=1  
            user=NAME_LIST[x+1]  
  
        if logged==0:  
            send_msg(sock, "You need to login to start a chat\n")  
  
    else:  
        temp2=""  
        for x in range(len(temp1)):  
            if x!=0:  
                if not temp2:  
                    temp2=str(temp1[x])  
                else:  
                    temp2+=" "  
                    temp2+=temp1[x]  
  
        broadcast(server_socket, sock, "["+user+"] : "+temp2+"\n")
```

Isi dari protokol sendall hampir sama dengan protokol send. Perbedaannya terdapat pada bagian akhir protokol yaitu fungsi yang dijalankan adalah fungsi broadcast. Fungsi ini memungkinkan client untuk mengirimkan pesan ke semua client yang terdaftar di server (kecuali client pengirim). Nantinya, client penerima akan menerima pesan yang dikirim oleh client pengirim.





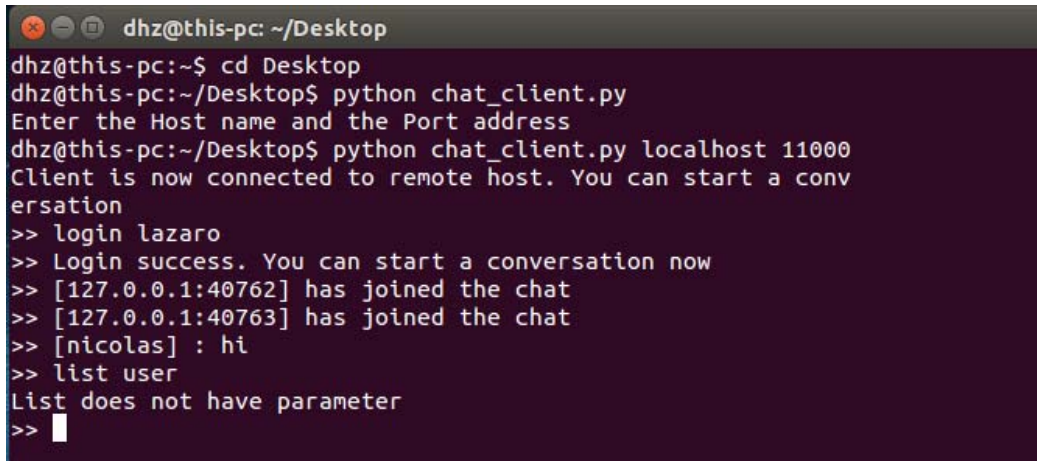
Gambar 8. Tampilan pesan yang di-sendall oleh client "lazaro" (kanan atas) pada layar client "alvin" (kiri bawah) dan "nicolas" (kanan bawah)

Berikut ini adalah isi dari fungsi broadcast.

```
def broadcast (server_socket, sock, message):
    for x in range (len(NAME_LIST)):
        if NAME_LIST[x] != server_socket and NAME_LIST[x] != sock
        and x%2==0 :
            try :
                NAME_LIST[x].send(message)
            except :
                NAME_LIST[x].close()
                if NAME_LIST[x] in SOCKET_LIST:
                    SOCKET_LIST.remove(NAME_LIST[x])
```

## List

Protokol list digunakan untuk menampilkan semua client yang terdaftar pada server. Artinya, protokol ini menampilkan username-username client yang tercantum dalam array NAME\_LIST. Syntax dari list adalah list, tanpa diikuti kata apapun di belakangnya. Jika terjadi kesalahan dalam penulisan perintah, akan ditampilkan pesan "List does not have parameter" pada layar client.

A terminal window titled 'dhz@this-pc: ~/Desktop' showing the execution of a Python chat client. The user enters 'python chat\_client.py' and then 'localhost 11000'. The client connects to a remote host. The user enters 'login lazaro', and the client responds with 'Login success. You can start a conversation now'. The user enters 'list user', and the client responds with 'List does not have parameter'.

```
dhz@this-pc:~/Desktop
dhz@this-pc:~/Desktop$ cd Desktop
dhz@this-pc:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
dhz@this-pc:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> login lazaro
>> Login success. You can start a conversation now
>> [127.0.0.1:40762] has joined the chat
>> [127.0.0.1:40763] has joined the chat
>> [nicolas] : hi
>> list user
List does not have parameter
>> 
```

Gambar 9. Tampilan pada layar client "lazaro" saat terjadi kesalahan penulisan perintah list

Berikut ini adalah potongan code untuk list.

```
elif temp1[0]=="list" :
    logged = 0
    for x in range (len(NAME_LIST)):
        if NAME_LIST[x]==sock:
            logged=1

    if logged==0:
        send_msg(sock, "You need to login to start a chat\n")

    else:
        temp2=""
        for x in range (len(NAME_LIST)):
            if x%2==1:
                temp2+=" "
                temp2+=str(NAME_LIST[x])

        send_msg(sock, "[List of User(s)] : "+temp2+"\n")
```

Isi dari protokol list hampir sama dengan protokol send dan send all. Perbedaannya adalah pada protokol list, temp2 diisi dengan username-username user yang terdaftar pada array NAME\_LIST. Seperti yang dijelaskan pada bagian login, array NAME\_LIST adalah array yang berisi alamat port dan username login client. Karena yang hendak ditampilkan adalah username client, isi dari array NAME\_LIST yang ditulis adalah isi yang terletak pada index ganjil. Ini terlihat pada syarat if yang digunakan yaitu `x%2==1`. Setelah

semua username dituliskan pada temp2, pesan akan dikirimkan ke client yang menuliskan perintah list dan ditampilkan pada layarnya.

```
andre@andre-X45U: ~/Desktop
andre@andre-X45U:~$ cd Desktop
andre@andre-X45U:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> login alvin
>> Login success. You can start a conversation now
>> send hello
Invalid Send Command
>> send lazaro hello
>> [lazaro] : test broadcast
>> list
>> [List of User(s)] : lazaro alvin nicolas
>>
```

Gambar 10. Tampilan pada layar client "alvin" setelah melakukan perintah list

## Who Am I

Protokol who am I (dalam chatting ditulis whoami) digunakan untuk menampilkan username client yang melakukan perintah whoami. Artinya, protokol ini menampilkan username client pengirim sesuai dengan yang tercantum dalam array NAME\_LIST. Syntax dari whoami adalah whoami, tanpa diikuti kata apapun di belakangnya. Jika terjadi kesalahan dalam penulisan perintah, akan ditampilkan pesan "Whoami does not have parameter" pada layar client.

```
dhz@this-pc: ~/Desktop
dhz@this-pc:~$ cd Desktop
dhz@this-pc:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
dhz@this-pc:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> login lazaro
>> Login success. You can start a conversation now
>> [127.0.0.1:40762] has joined the chat
>> [127.0.0.1:40763] has joined the chat
>> [nicolas] : hi
>> list user
List does not have parameter
>> whoami server?
Whoami does not have parameter
>>
```

Gambar 11. Tampilan pada client "lazaro" saat terjadi kesalahan penulisan perintah whoami

Berikut ini adalah potongan code untuk whoami.

```
elif temp1[0]=="whoami" :
    g = 0
    for name in range (len(NAME_LIST)):
```

```

if NAME_LIST[name]==sock:

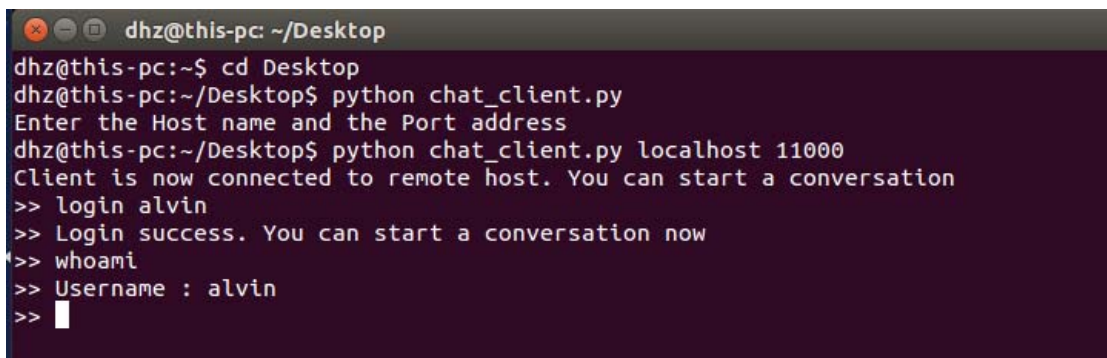
    g = 1
    send_msg(sock, "Username :
"+str(NAME_LIST[name+1])+"\n")

if g==0:

    send_msg(sock, "You haven't login\n")

```

Dalam protokol whoami, terdapat variable g yang berfungsi sebagai status apakah username client tersebut sudah terdaftar pada array NAME\_LIST atau belum. Pada awalnya, g akan di-set menjadi 0. Kemudian dilakukan pengecekan username client pada array NAME\_LIST. Jika username terdaftar pada NAME\_LIST, status g akan diubah menjadi 1 dan dikirimkan pesan “Username: username client” ke client yang melakukan perintah whoami.



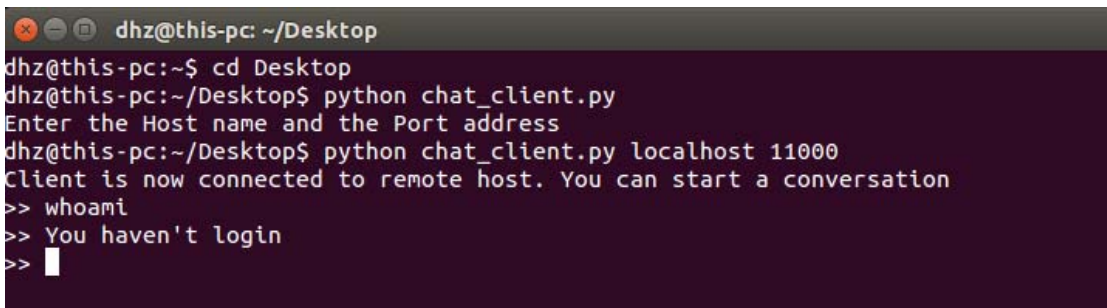
```

dhz@this-pc: ~/Desktop
dhz@this-pc:~$ cd Desktop
dhz@this-pc:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
dhz@this-pc:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> login alvin
>> Login success. You can start a conversation now
>> whoami
>> Username : alvin
>> 

```

Gambar 12. Tampilan pada layar client "alvin" setelah melakukan perintah whoami

Akan tetapi, jika username client tidak ditemukan pada array NAME\_LIST maka status g tetap bernilai 0 dan dikirimkan pesan “You haven’t login” kepada client yang melakukan perintah whoami.



```

dhz@this-pc: ~/Desktop
dhz@this-pc:~$ cd Desktop
dhz@this-pc:~/Desktop$ python chat_client.py
Enter the Host name and the Port address
dhz@this-pc:~/Desktop$ python chat_client.py localhost 11000
Client is now connected to remote host. You can start a conversation
>> whoami
>> You haven't login
>> 

```

Gambar 13. Tampilan pada layar client jika melakukan perintah whoami saat belum login