**DOKUMENTASI**

**PROGRAM CHAT CLIENT-SERVER**



Anugerah Yulindra Satyaji (5113100186)

Pemrograman Jaringan B

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**FAKULTAS TEKNOLOGI INFORMASI**

**JURUSAN TEKNIK KOMPUTER INFORMATIKA**

**SURABAYA**

**2015**

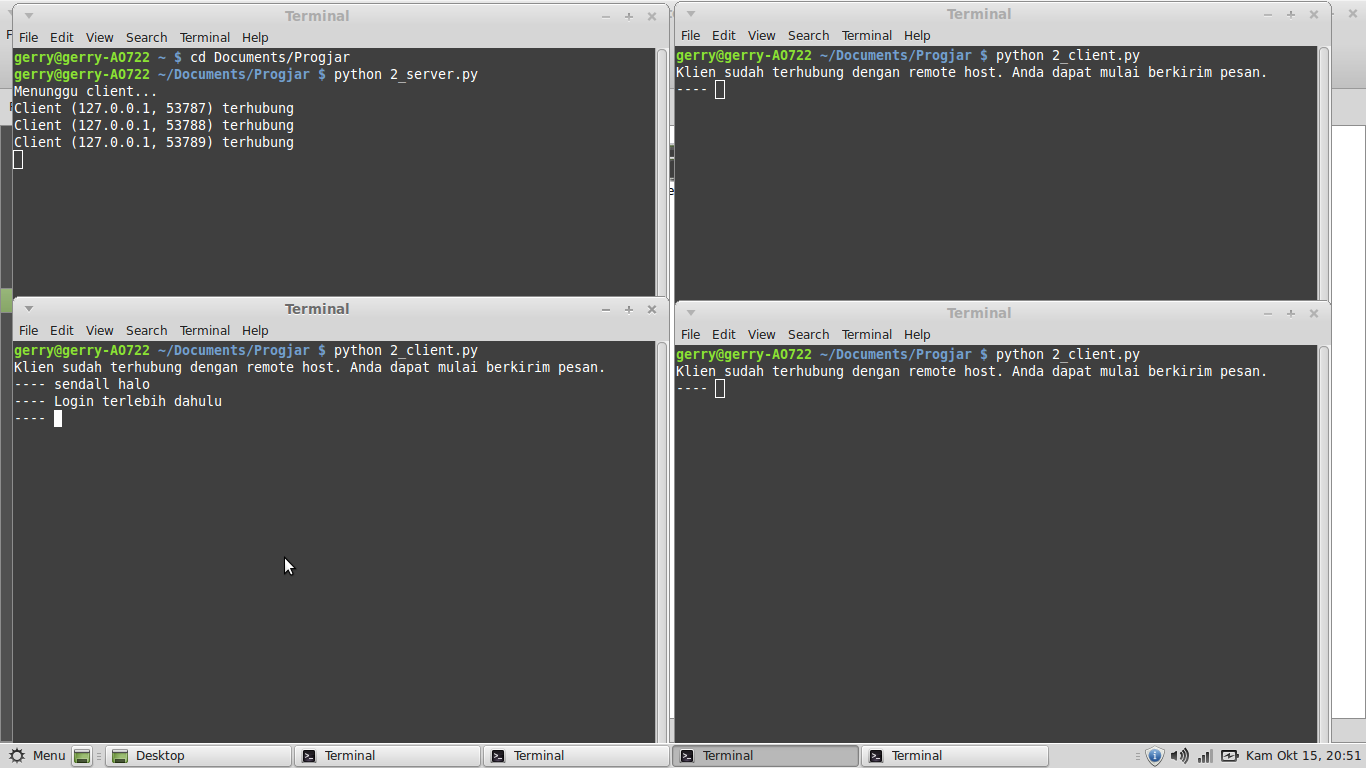
Pada program *chat* ini, saya menggunakan protokol TCP/IP dibantu dengan empat protokol lain dalam menjalankan fungsinya.

Pertama, sebelum kita dapat mulai menjalankan segala protokol yang ada dalam program, terlebih dahulu kita harus mengkoneksikan/menghubungkan antara server dengan klien. Agar dapat terhubung, saya mengatur host menjadi *localhost* dan port menjadi 8080 pada server dan klien.

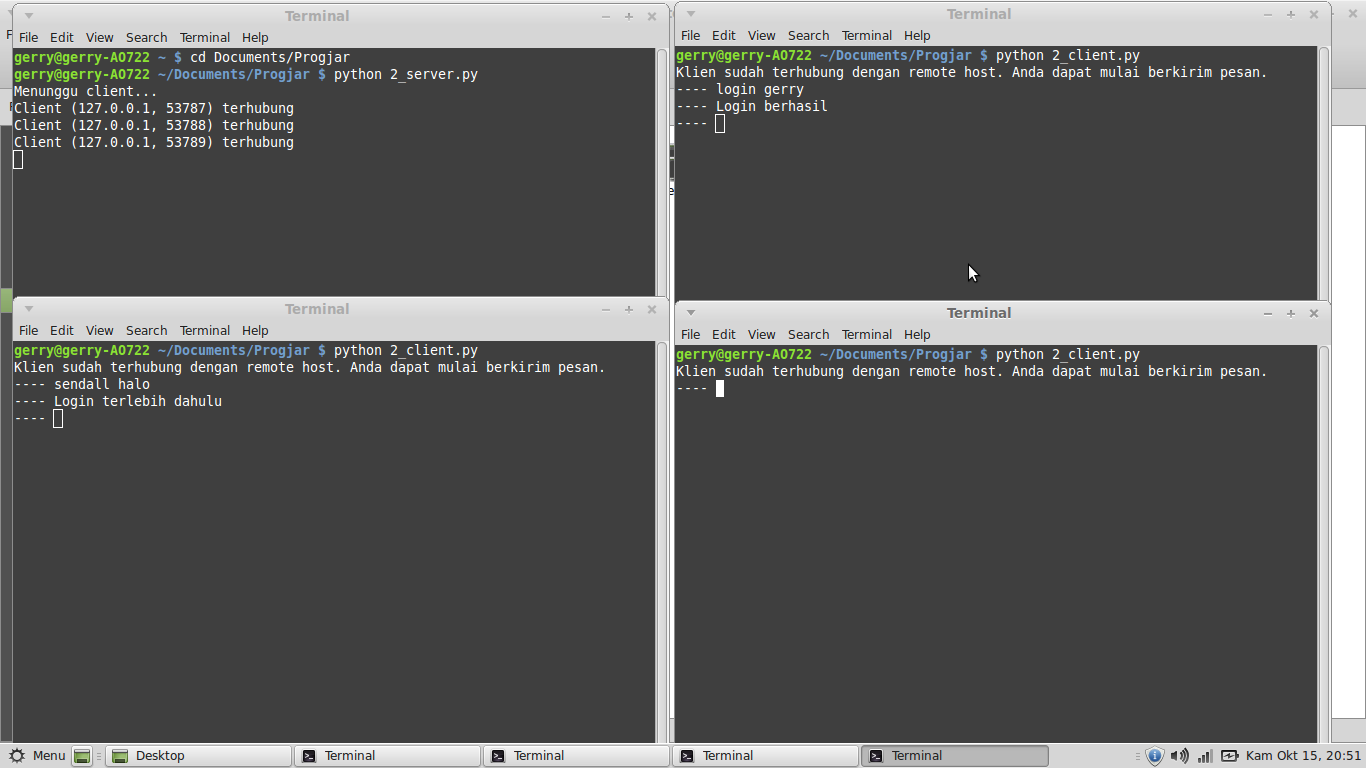
Dalam penggunaannya, kita jalankan terlebih dahulu server, disusul dengan klien (dalam hal ini, saya membuka tiga klien). Setelah semua terhubung, baru kita bisa lanjut ke dalam program.

1. Login

Terlebih dahulu, kita harus *login* untuk dapat menjalankan program tersebut. Untuk *login*, Anda cukup mengetikkan dengan format “*login(spasi)nama”*. Jika nama belum dipakai sama sekali maka nama tersebut akan disimpan dalam *array*. Namun, jika Anda *login* menggunakan nama yang sama, maka program akan mengeluarkan peringatan “Username sudah ada yang punya”. Sama halnya jika *login* tanpa adanya nama dan menggunakan spasi, maka program akan mengeluarkan peringatan “Login butuh username” dan “Username cacat”.



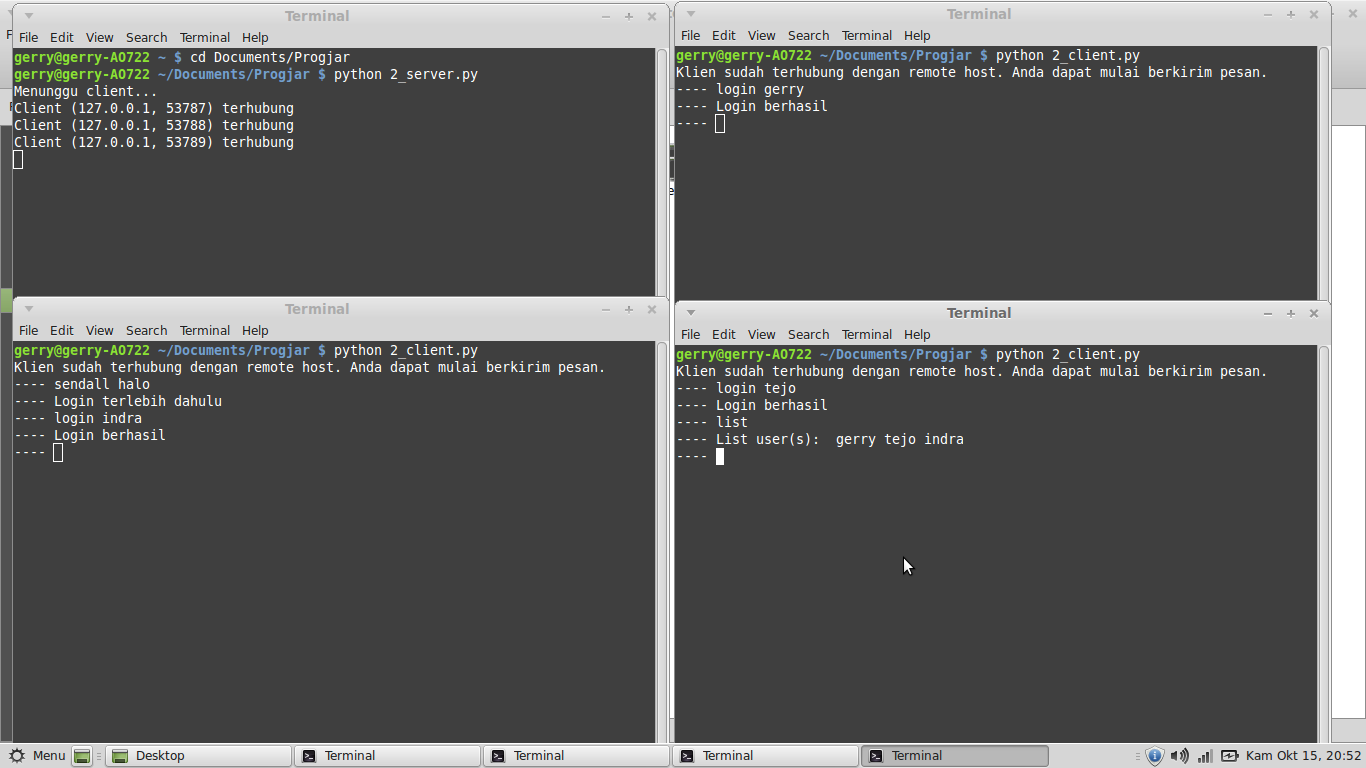
**Gambar 1.1** Gagal melakukan perintah lain karena belum *login*



**Gambar 1.2** Berhasil *login*

1. List

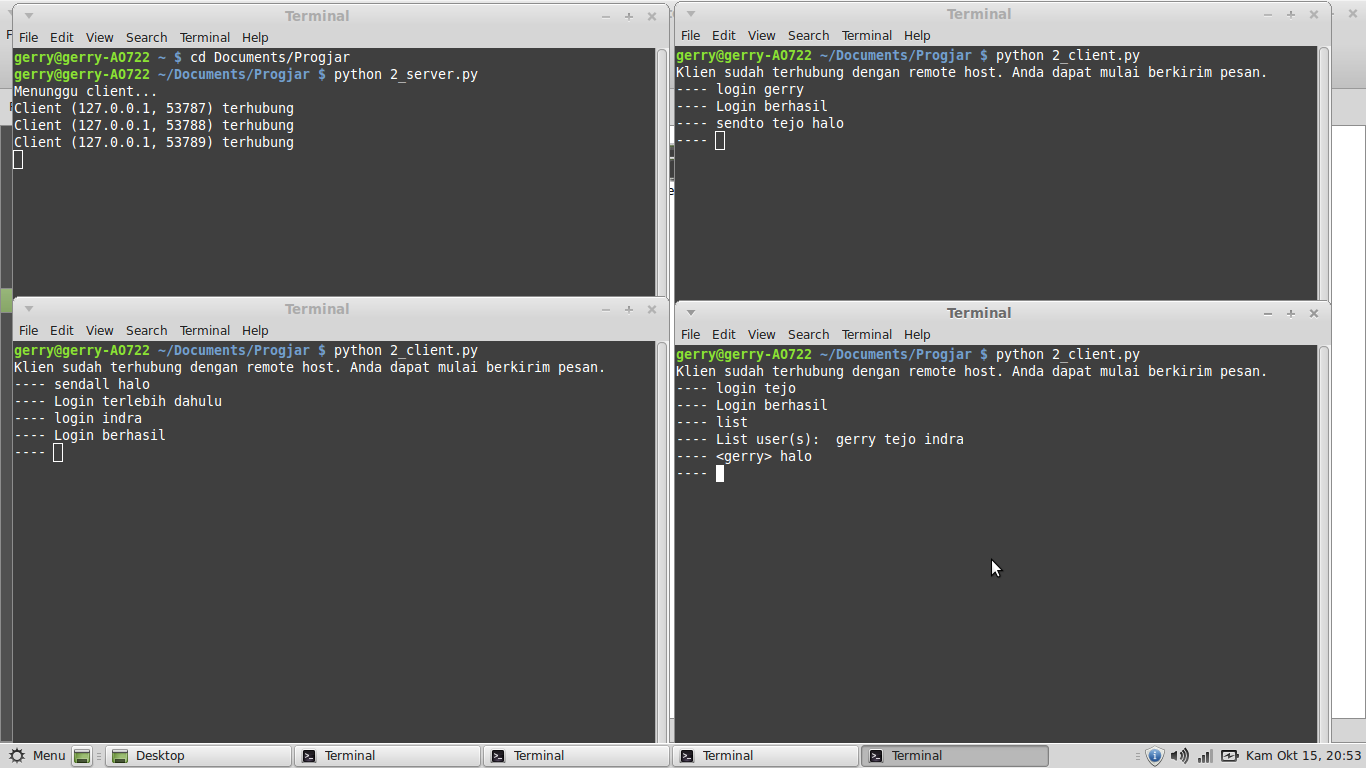
Selanjutnya, ada fitur *list*. Fungsi *list* adalah untuk mengetahui siapa saja *user* yang sedang aktif dalam ruang percakapan yang sedang berjalan. Format untuk menggunakan fitur tersebut adalah dengan mengetikkan “*list*” saja. Perintah ini tidak memiliki parameter. Untuk menampilkan setiap *user* yang sedang aktif, maka dicocokkan apakah daftar nama yang telah disimpan di awal sama dengan yang di socket. Pencocokkan tersebut berulang sebanyak jumlah daftar nama yang ada pada socket. Jika telah selesai, maka nama-nama yang telah dicocokkan tersebut akan dikirim balik untuk bisa ditampilkan pada perintah *list.*

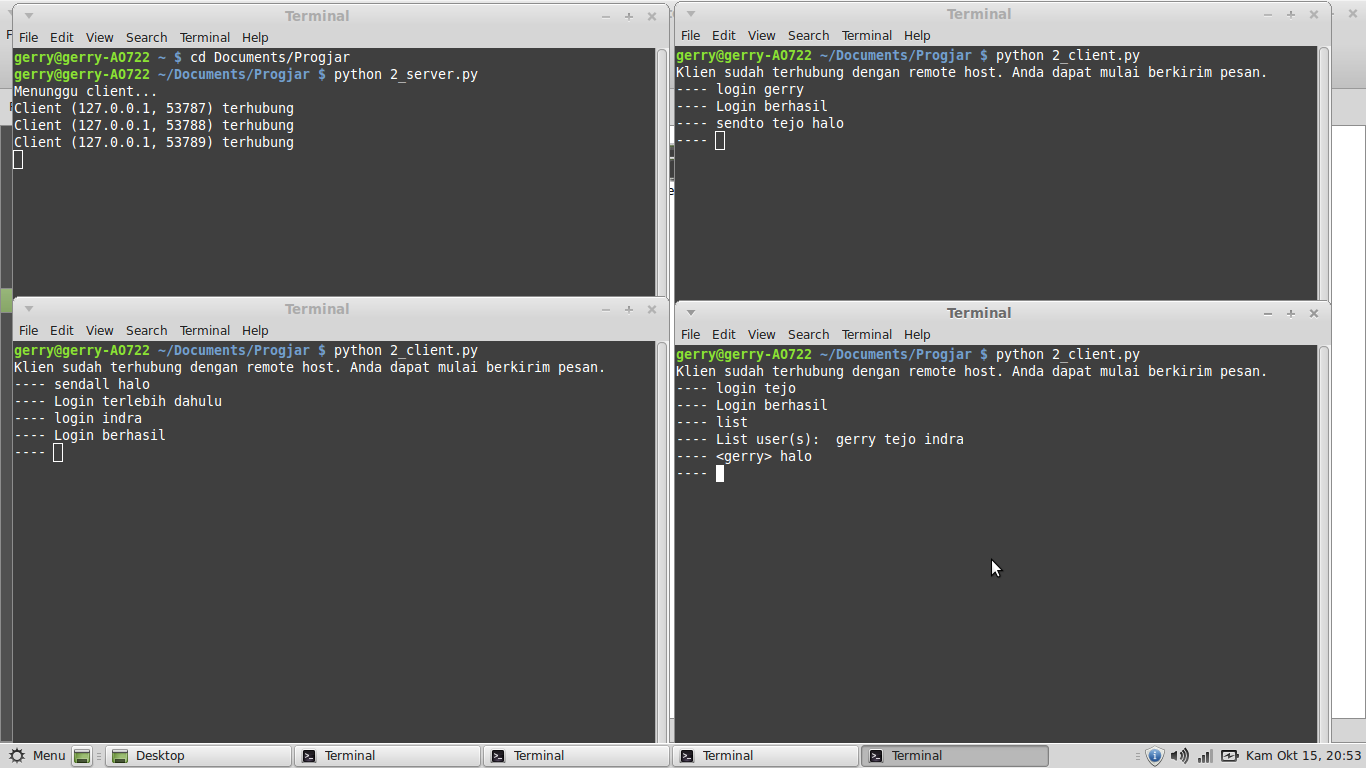


**Gambar 2.1** Hasil dari menjalankan *list*

1. Sendto

Program ini juga memiliki fitur lagi dalam mengirim pesan ke *user* lain, yakni *sendto*. Fitur *sendto* sendiri berfungsi untuk berkirim pesan secara spesifik kepada satu *user* saja, tidak bisa lebih daripada satu *user.* Untuk menjalankan fitur *sendto*, cukup ketik “*sendto(spasi)nama(spasi)pesan*”. Bisa kita lihat bahwa terdapat dua parameter agar fungsi *sendto* dapat berjalan, yakni nama klien yang ingin kita kirim pesan dan tentu saja pesan itu sendiri. Tentunya, setiap parameter dan juga perintah *sendto* itu sendiri dipisah sendiri-sendiri agar dapat dicocokkan, misalnya kepada siapa pesan tersebut akan dikirim? *User* tersebut haruslah sedang aktif dan cocok dengan daftar *user* yang telah disimpan pada perintah *login* diawal tadi. Dan dengan otomatis akan memanggil fungsi *send\_msg().*

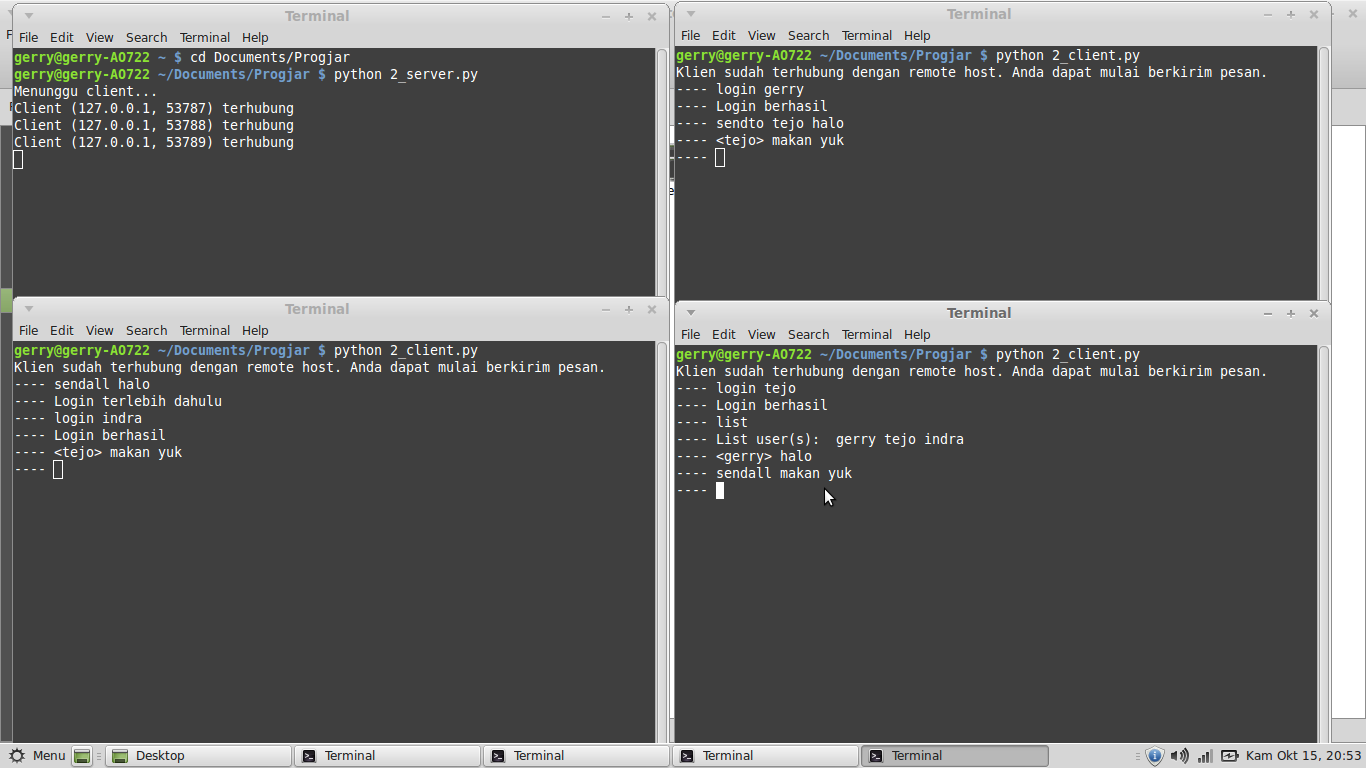


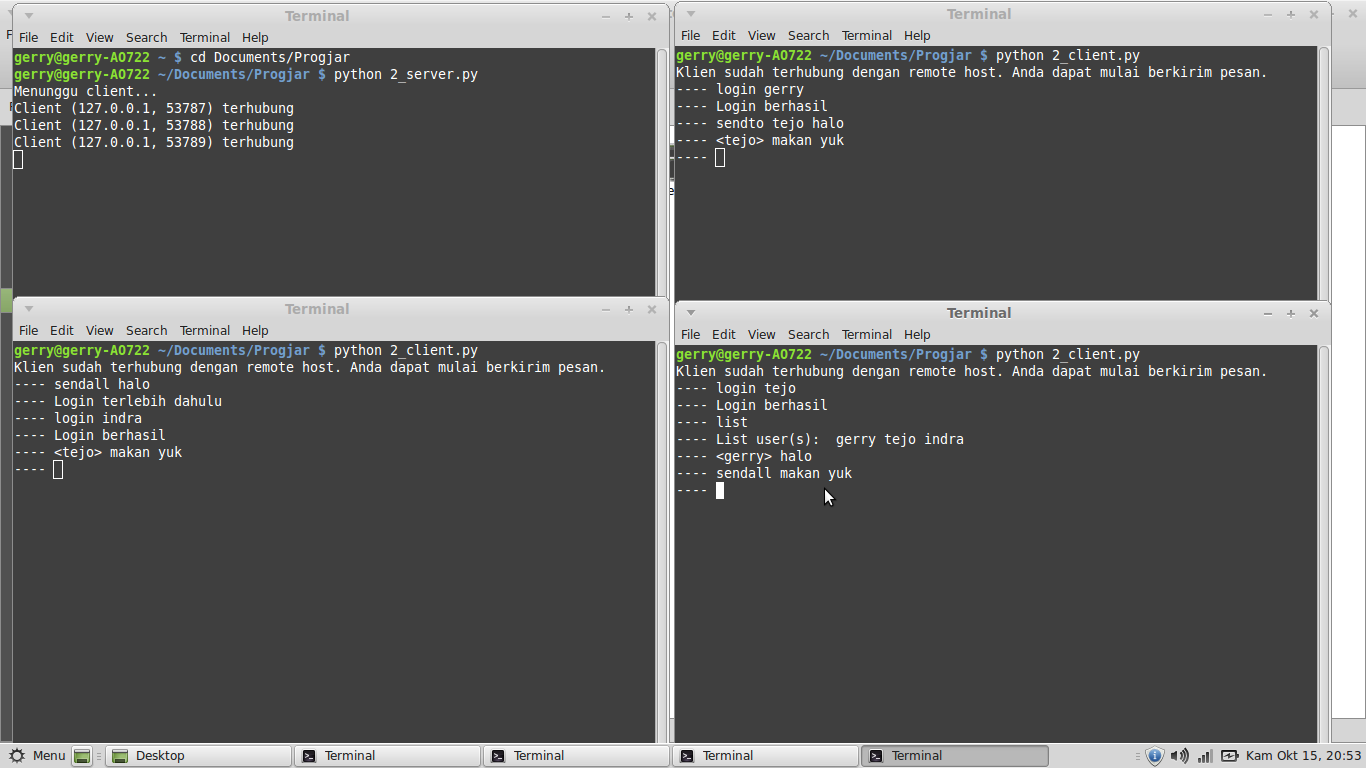


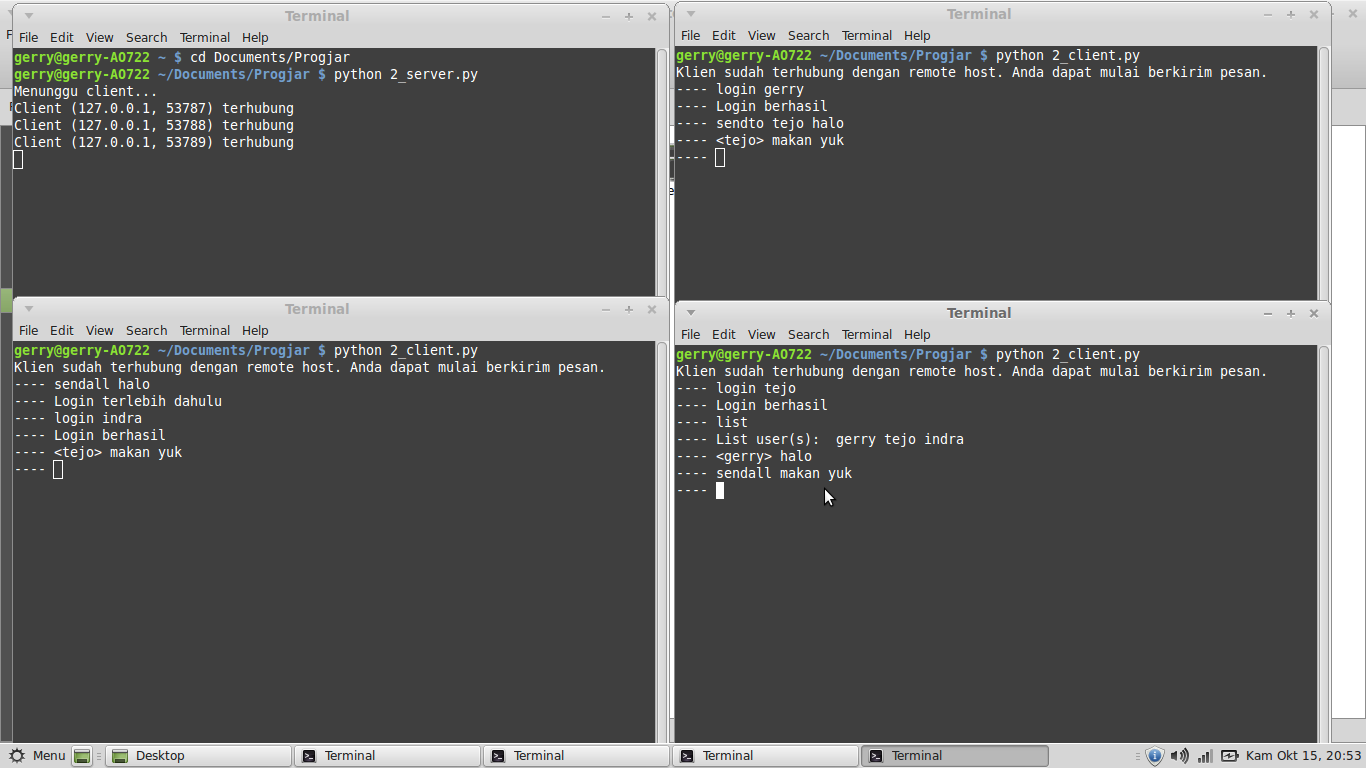
**Gambar 3.1** Hasil dari menjalankan *sendto* antara *user* gerry dan tejo

1. Sendall

Terakhir, merupakan fitur *sendall*. Fitur ini berfungsi untuk berkirim pesan, namun ke semua *user* yang sedang aktif. Cara kerjanya adalah cukup dengan mengetikkan “*sendall(spasi)pesan*”. Alurnya hampir sama seperti perintah sebelumnya, namun bedanya ketika perintah tersebut diketik dan dijalankan, maka fungsi *broadcast()-*lah yang akan bekerja. Pesan yang terkirim akan dikirim ke semua *user* yang aktif berdasarkan dengan daftar nama *user* yang telah disimpan pada perintah *login* di awal.



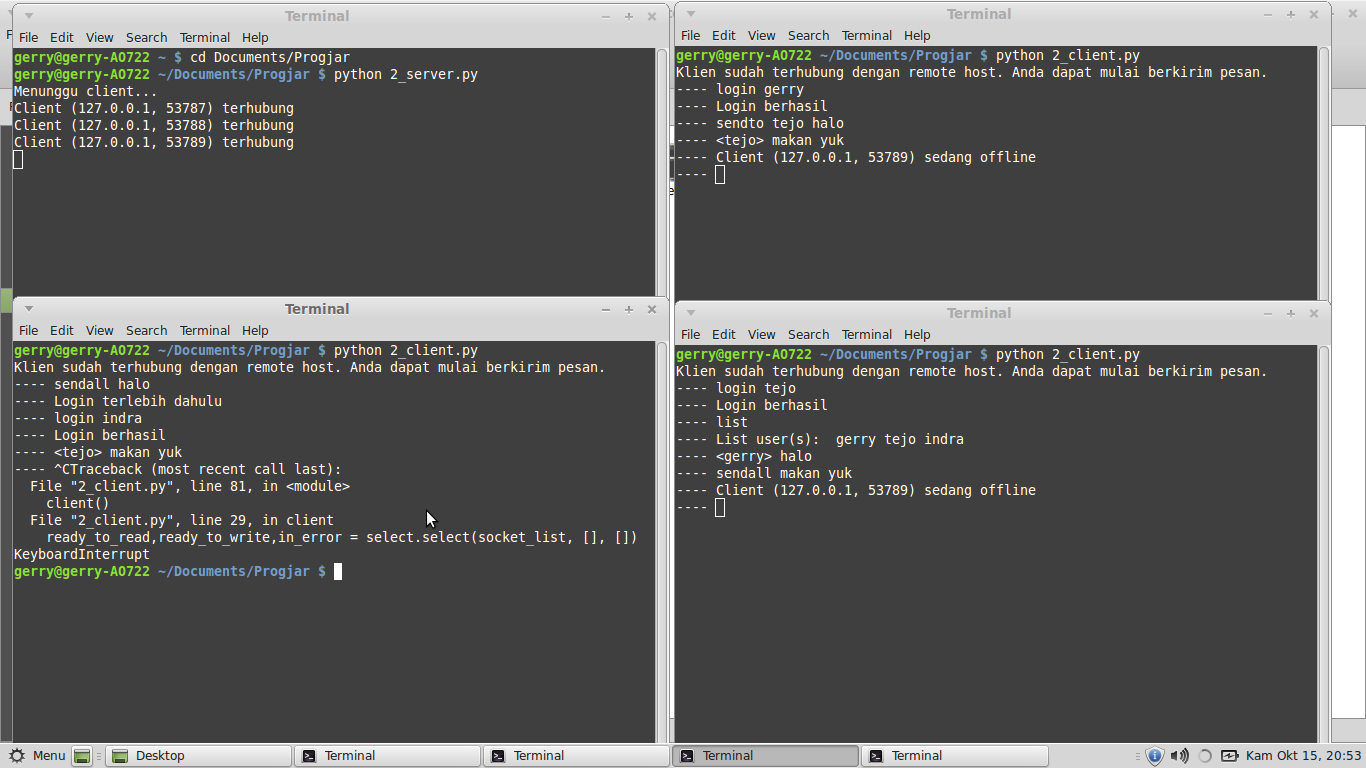


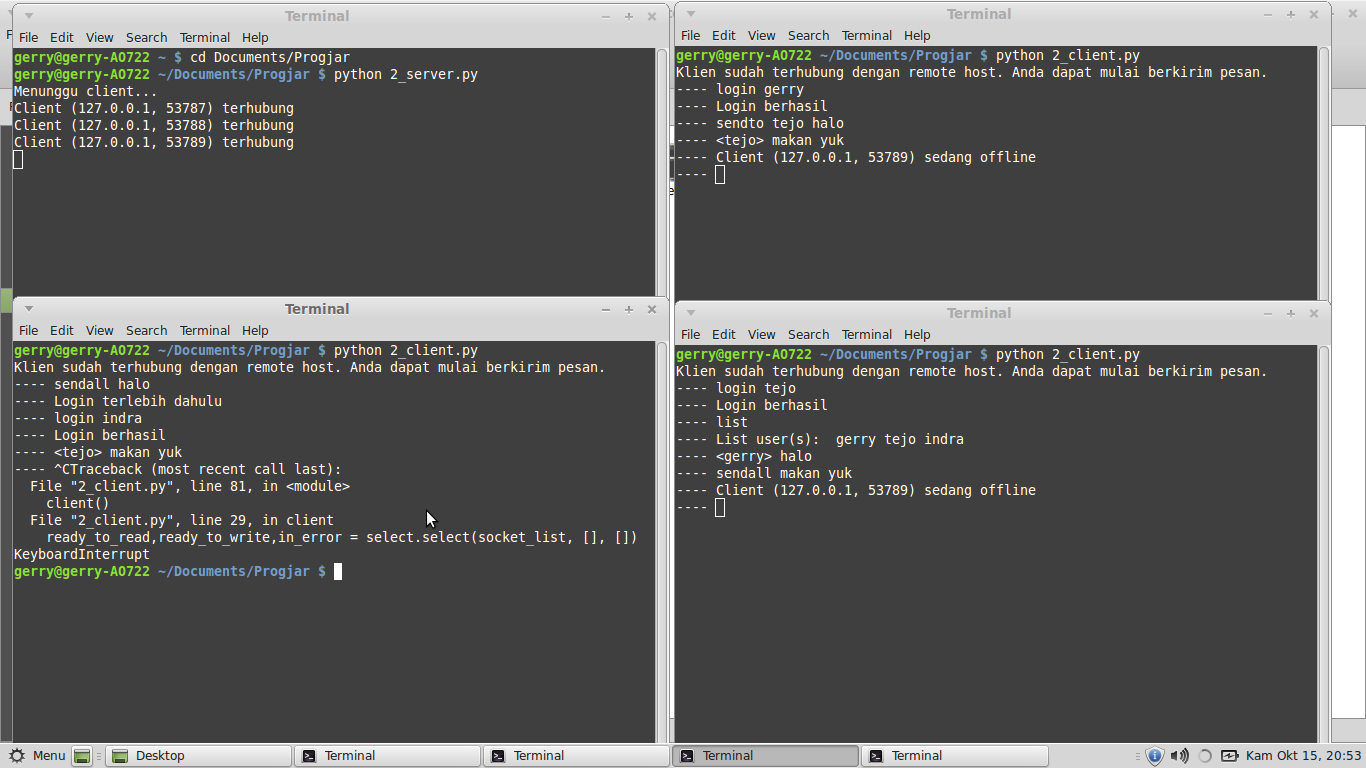


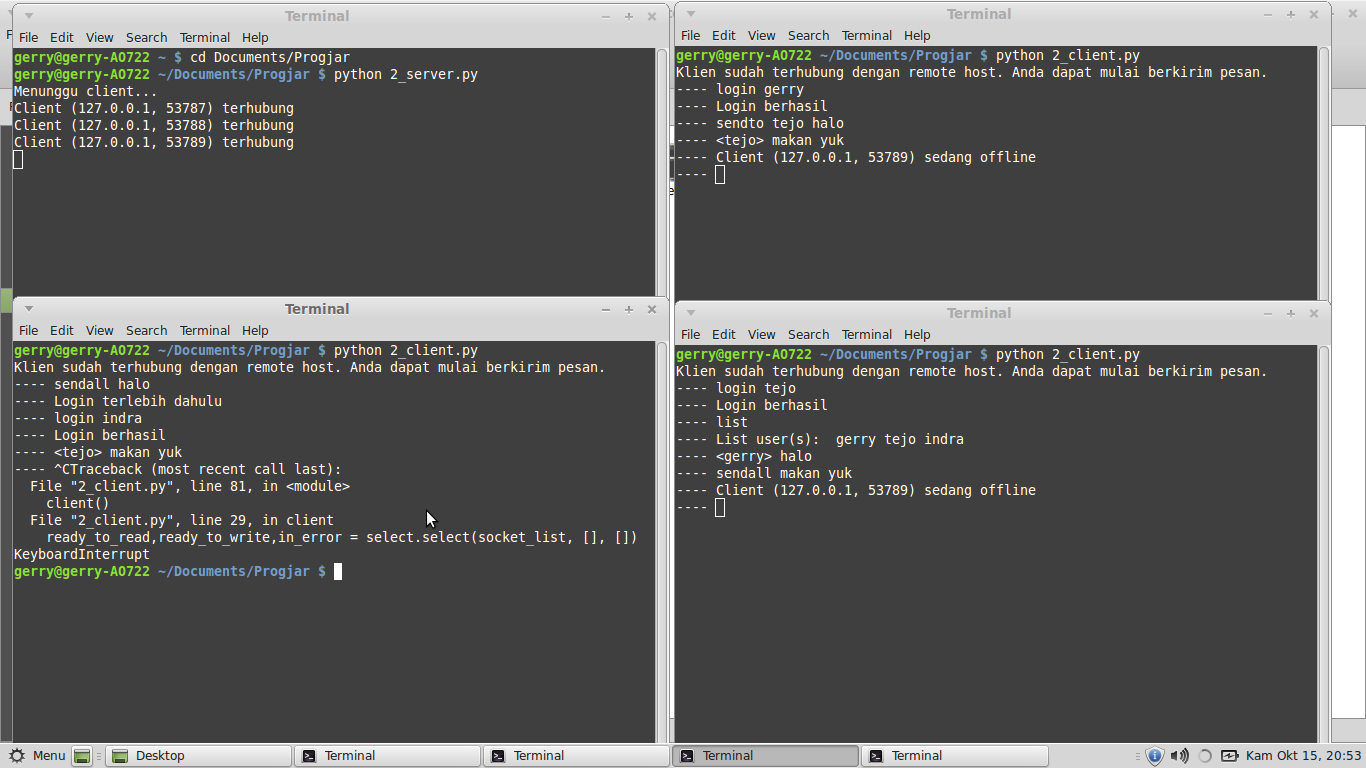
**Gambar 4.1** Hasil dari menjalankan *sendall* yang dikirim dari tejo ke gerry dan indra

(Tambahan)

Ketika salah satu *user ada* yang keluar dari ruang *chat.*







.