

# Pemrograman Socket dengan Python

**Rikih Gunawan**

*rikih.gunawan@gmail.com*

## ***Lisensi Dokumen:***

*Copyright © 2003-2006 IlmuKomputer.Com*

*Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.*

## **Pendahuluan**

Untuk pembuatan aplikasi jaringan, Python menyediakan modul-modul untuk mempermudah para pembuat program, sehingga para pemrogram lebih memfokuskan diri pada cara kerja dari programnya tersebut dan yang akan dibahas adalah penggunaan modul socket. Dalam penulisan ini penulis hanya membahas penggunaan socket dasar di Python dan berasumsi bahwa Anda minimal telah mengerti konsep jaringan dan dapat menggunakan perintah-perintah dasar Python.

## **Model Client-Server (*Client-Server Model*)**

Aplikasi jaringan yang sering dibuat diasumsikan sebagai *client* dan sisi lainnya *server*. Tujuannya adalah agar *server* dapat memberikan layanan terhadap *client*-nya. Kita dapat mengkategorikan *server* dalam dua kelas : *Iterative* atau *Concurrent Server*. Untuk *iterative server* akan terus melaksanakan langkah-langkah berikut :

1. Tunggu sampai permintaan (*request*) dari *client* tiba.
2. Proses permintaan *client*.
3. Kirimkan respon (*response*) kembali ke *client* yang telah mengirim permintaan (*request*).
4. Kembali ke langkah 1

Masalah pada *iterative server* adalah ketika sedang menjalankan langkah 2. Selama waktu tersebut tidak ada *client* yang dilayani. Sedangkan untuk *concurrent server* akan melaksanakan langkah-langkah berikut :

1. Tunggu sampai permintaan (*request*) dari *client* tiba.

2. Jalankan *server* baru untuk menangani permintaan (*request*) dari *client*. Ini dapat dilakukan dengan membuat proses baru atau menggunakan *thread* tergantung dari dukungan sistem operasi yang digunakan. Setelah permintaan dilayani, *server* baru tersebut akan mati.
3. Kembali ke langkah 1.

## Penggunaan Dasar Socket

Python hanya menggunakan dua domain komunikasi, yaitu : UNIX (`AF_UNIX`) dan Internet (`AF_INET`) domain. Pengalamatan pada UNIX domain direpresentasikan sebagai *string*, dinamakan dalam lokal path: contoh `/tmp/sock`. Sedangkan pengalamatan Internet domain direpresentasikan sebagai *tuple* (`host, port`), dimana *host* merupakan *string* yang merepresentasikan nama *host* internet yang sah (*hostname*), misalnya : `darkstar.drslump.net` atau berupa IP address dalam notasi *dotted decimal*, misalnya : `192.168.1.1`. Dan port merupakan nomor port yang sah antara 1 sampai 65535. Tetapi dalam keluarga UNIX penggunaan port di bawah 1024 memerlukan akses *root privileges*. Sebelum menggunakan modul socket dalam Python, maka modul socket harus terlebih dahulu diimport. Berikut contohnya :

```
#!/usr/bin/env python
#Mengimport modul socket
import socket
# Mengimport seluruh konstanta, data, dan method
from socket import *
# Mengimport konstanta
from socket import AF_INET, SOCK_STREAM
```

## Membuat Socket (*Creating*)

Socket dibuat melalui pemanggilan `socket(family, type[, Proto])`. Untuk lebih jelasnya dapat dilihat pada tabel 1 dan tabel 2 berikut ini :

**Tabel 1** Konstanta Keluarga (Family) Protokol

Family	Penjelasan
AF_UNIX	Unix Domain Protocol
AF_INET	IPv4 Protocol
AF_INET6	IPv6 Protocol

**Tabel 2** Konstanta Type Socket

Type	Penjelasan
SOCK_STREAM	Stream Socket (TCP)
SOCK_DGRAM	Datagram Socket (UDP)
SOCK_RAW	Raw Socket
SOCK_RDM	-
SOCK_SEQPACKET	-

Untuk proto bersifat opsional dan biasanya bernilai 0. Untuk membuat socket stream (TCP) internet domain digunakan *statement* berikut :

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Jika SOCK\_STREAM diganti dengan SOCK\_DGRAM berarti membuat socket datagram (UDP). Kemudian untuk membuat socket stream dalam UNIX domain :

```
sock = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
```

## Menghubungkan Socket (*Connecting*)

Sebuah *server* dari sudut pandang kita adalah sebuah proses yang mendengarkan (*listen*) pada port tertentu. Ketika proses lain ingin berhubungan dengan *server* atau menggunakan layanan *server*, maka proses harus terhubung dengan alamat dan nomor port tertentu yang dispesifikasikan oleh *server*. Ini dilakukan dengan memanggil metode `socket connect(address)`, dimana `address` adalah sebuah *tuple* (`host, port`) untuk Internet domain dan `pathname` untuk UNIX domain. Berikut contohnya :

```
sock.connect (('localhost',12345)) atau  
sock.connect (('192.168.1.1',12345))
```

Sedangkan untuk UNIX domain,

```
sock.connect ('/tmp/sock') #Koneksi ke file socket
```

## Mengikatkan Socket ke Port (*Binding*)

Setelah socket berhasil dibuat, maka Python akan mengembalikan sebuah *socket descriptor*. Sebelum digunakan, maka socket harus diikatkan (*binding*) ke alamat dan nomor port yang sesuai agar proses lain dapat ditujukan ke socket. Berikut ini contoh untuk *binding* socket pada internet domain :

```
sock.bind(('localhost',12345)) atau  
sock.bind(('192.168.1.1',12345))
```

Sedangkan untuk mengikatkan (*binding*) socket pada UNIX domain digunakan :

```
sock.bind('/tmp/sock') #/tmp/sock merupakan file socket
```

Perintah di atas akan membuat file pipe `/tmp/sock` yang dapat digunakan untuk berkomunikasi antara *server* dan *client*.

## Mendengarkan Koneksi (*Listening*)

Setelah socket diikatkan (*bind*), langkah selanjutnya adalah memanggil method `listen(queue)`. Perintah ini menginstruksikan socket untuk *listen* pada port-port yang telah diikatkan (*bind*), dan `queue` merupakan sebuah integer yang merepresentasikan maksimum antrian koneksi, berikut contoh penggunaannya :

```
sock.listen(5) #Mendengarkan koneksi dengan maksimum
               antrian sebanyak 5
```

## Menerima Koneksi (*Accepting*)

Untuk menerima koneksi dari permintaan (*request*) *client* pada koneksi yang menggunakan socket stream (TCP). Method yang digunakan `accept()`, berikut contoh penggunaannya :

```
sock.accept() #Menerima koneksi
```

*Statement* di atas akan mengembalikan sebuah *tuple* (`conn, address`) dimana `conn` adalah objek socket baru yang berguna untuk mengirim dan menerima data dari koneksi, dan `address` merupakan alamat dari *client*.

## Mengirim Data ke Koneksi (*Sending*)

Menerima koneksi tidak akan berarti tanpa digunakan untuk mengirim dan menerima data. Oleh karena itu digunakan method `send(string)` untuk socket stream (TCP) dan `sendto(string, address)` untuk socket datagram (UDP). Berikut ini penggunaannya untuk socket stream.

```
sock.send('ini pesan dari server')
```

Sedangkan untuk socket datagram digunakan :

```
sock.sendto('pesan dari server' , ('192.168.1.1' , 12345))
```

## Menerima Data Dari Koneksi (*Receiving*)

Untuk menerima data yang dikirim dari *server* digunakan method `recv(bufsize)` untuk socket stream dan `recvfrom(bufsize)`. Berikut ini penggunaannya untuk socket stream :

```
sock.recv(1024) #Menerima data sebesar 1024 byte
```

*Statement* di atas akan mengembalikan data yang dikirimkan oleh *client*. Sedangkan untuk socket datagram :

```
sock.recvfrom(1024) #Menerima data sebesar 1024 byte
```

*Statement* di atas akan mengembalikan dua buah field yaitu *data*, *address*.

## Menutup Koneksi (*Closing*)

Untuk menutup koneksi yang telah dibuat digunakan method `close(s)`. Berikut penggunaannya :

```
sock.close() #Menutup koneksi
```

## Belajar melalui contoh

Pada bagian ini akan dibahas mengenai contoh-contoh aplikasi jaringan sederhana. Aplikasi-aplikasi yang dibahas masih memiliki banyak kekurangan (dalam hal logika, teknik dan lain), tetapi bukan itu yang ingin penulis sampaikan, penulis berharap contoh program yang dibahas dapat menjadi sarana belajar bagi Anda dan dapat dikembangkan lebih baik lagi. Program yang dibawah ini dibuat dan dijalankan pada Python 2.2.2 atau lebih dan tidak menutup kemungkinan dapat dijalankan di bawah versi 2.2.2. Untuk mencoba menjalankannya gunakan 2 buah terminal/console untuk server dan client. Program yang akan dibahas ada 5 buah program, yaitu :

### 1. Program mengirim pesan [single.py]

```
#!/usr/bin/env python

# import modul socket dan sys
import socket, sys

# Untuk welcome string
welstr = '\n\rSelamat datang di dr_slump chat server \n\r\
Powered by dr_slump Technology\n\r\n\r\
User Access Verification\n\r\
Password: '

class Net:
    def __init__(self):
        # Cek argumen, jika tidak sama dengan 3
        # tampilkan cara penggunaan
        if len(sys.argv) != 3:
            print "Usage: " + sys.argv[0] + " <hostip> <port>"
            sys.exit(1)
        else:
            self.HOST = sys.argv[1] # Set nilai variabel dari
            self.PORT = int(sys.argv[2]) # parameter yang diberikan
            self.prompt = 'chat> ' # prompt yang akan
ditampilkan

    def Create(self):
        try:
            # Buat socket INET dengan protocol TCP
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        except:
            print "Buat socket error..."
        else:
            # Binding ip dan port
            s.bind((self.HOST, self.PORT))

            # Mendengarkan koneksi
            s.listen(2)

            # Menerima koneksi yang datang
            koneksi, alamat = s.accept()

            # Setelah koneksi diterima, server mengirim pesan
```

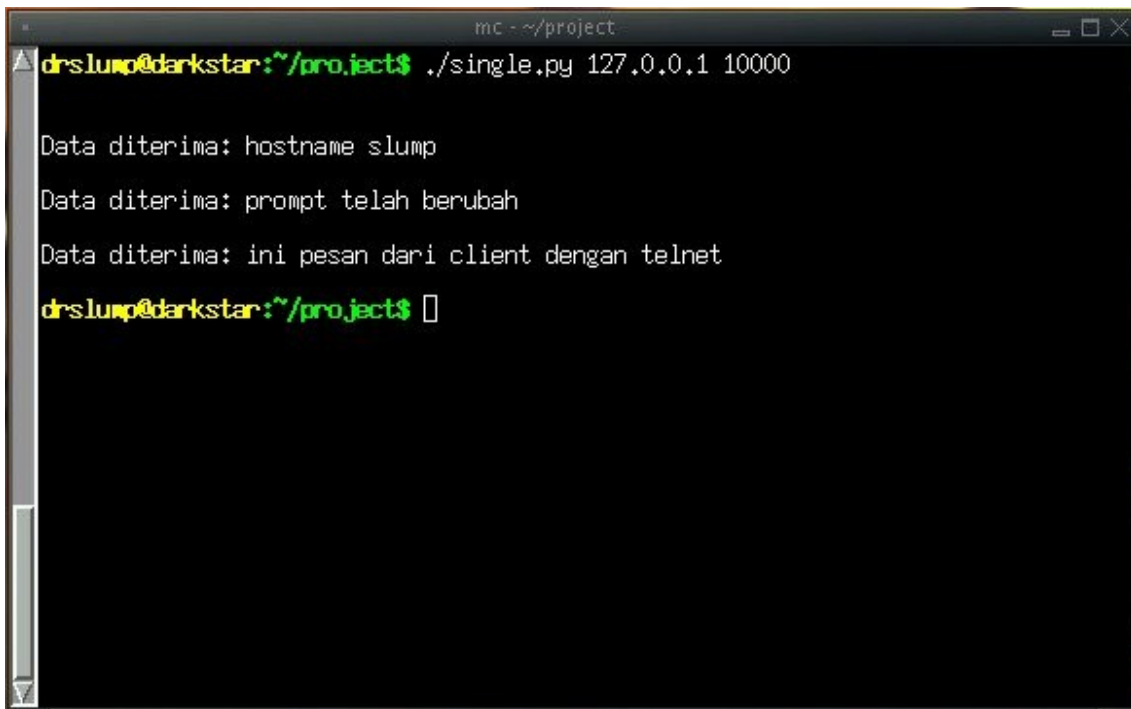
```
# selamat datang ke client
koneksi.send(welstr)
stat=0 # Flag untuk status koneksi
      # 0=tdk terkoneksi, 1=terkoneksi
while 1:
    # Terima data
    data = koneksi.recv(100)
    if not data: break
    if stat==0:
        if data.strip() == "password":
            stat=1
            isi=self.prompt
            koneksi.send("Anda      berhasil      login      ke
server\n\nr")
        else:
            isi = 'Password: '
    else:
        if data[:8] == 'hostname':
            host = data.split(' ')[1]
            self.prompt = host.strip() + '> '
            isi=self.prompt
        if data.strip() in ['keluar']:
            koneksi.close()
            break
    print 'Data diterima: ' + str(data)
    koneksi.send(isi)
s.close()

net = Net()
net.Create()
```

### Penjelasan Program [single.py]

Program diatas merupakan contoh sederhana program jaringan yang menggunakan Internet Domain dengan protocol TCP. Program ini hanya terdiri dari satu program saja dan berfungsi sebagai Server. Sedangkan untuk clientnya menggunakan aplikasi Telnet yang ada pada sistem Anda. Pada saat program dijalankan dan client melakukan koneksi dengan Telnet maka server akan meminta verifikasi password. Jika client memasukkan password "password" maka server akan mengirimkan pesan bahwa client telah terkoneksi dan sebaliknya jika client salah memasukkan password server akan terus meminta password. Perintah-perintah yang dapat digunakan hanya perintah hostname [prompt] untuk mengganti string prompt, dan keluar untuk memutuskan koneksi dan keluar dari program.

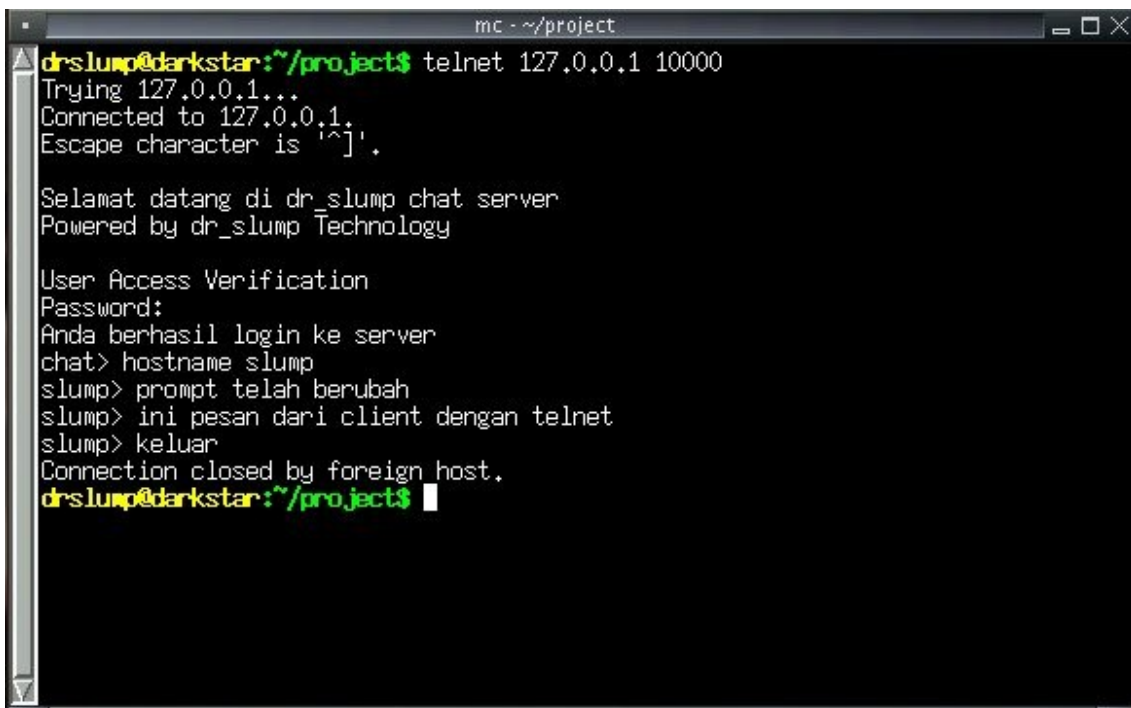
### Output Program [single.py]



```
mc - ~/project
drslump@darkstar:~/project$ ./single.py 127.0.0.1 10000

Data diterima: hostname slump
Data diterima: prompt telah berubah
Data diterima: ini pesan dari client dengan telnet
drslump@darkstar:~/project$
```

Menjalankan Server



```
mc - ~/project
drslump@darkstar:~/project$ telnet 127.0.0.1 10000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

Selamat datang di dr_slump chat server
Powered by dr_slump Technology

User Access Verification
Password:
Anda berhasil login ke server
chat> hostname slump
slump> prompt telah berubah
slump> ini pesan dari client dengan telnet
slump> keluar
Connection closed by foreign host.
drslump@darkstar:~/project$
```

Koneksi ke Server dengan Telnet

## 2. Program Chatting dengan socket Internet menggunakan protocol TCP

### Program Server [pySChat.py]

```
#!/usr/bin/env python

from socket import *
import sys

class TCPServer:
    def __init__(self):
        if len(sys.argv) != 3:
            print 'Penggunaan: '+sys.argv[0]+' [ip_address] [nomor_port]'
            sys.exit(1)
        else:
            self.HOST = sys.argv[1]
            self.PORT = int(sys.argv[2])

    def Create(self):
        try:
            self.sockTCP = socket(AF_INET,SOCK_STREAM) # Buat socket
TCP
            self.sockTCP.bind((self.HOST, self.PORT)) # Binding port
            self.sockTCP.listen(1) # Listening
        except:
            print 'Socket error [ip dan port harus valid]'
            sys.exit()
        else:
            print 'Server Message [tekan Ctrl-C untuk keluar]'
            print '-----'
            print 'Mendengarkan pada port ' + str(self.PORT)

    def Accept(self):
        koneksi, alamat = self.sockTCP.accept() # Terima koneksi
        print 'Koneksi dari ' + str(alamat)
        while 1: # Lakukan perulangan selama true
            data = koneksi.recv(1024)
            print 'Pesan dari client >> ' + str(data) # Cetak pesan
client
            if not data: break
            if len(data) > 1: # Kirim pesan
balik
                koneksi.send('[' + str(data) + '] sudah diterima
server.')
```

### Penjelasan Program Server [pySChat.py]

Program diatas merupakan program yang berfungsi sebagai server untuk chatting. Program ini merupakan server yang bertipe Iterative server, yang berarti hanya dapat menerima dari satu client/koneksi saja.

### Output Program [pySChat.py]



```
mc - ~/project/ftp
drslump@darkstar:~/project/chat$ ./pySChat.py 127.0.0.1 2000
Server Message [tekan Ctrl-C untuk keluar]
-----
Mendengarkan pada port 2000
Koneksi dari ('127.0.0.1', 1028)
Pesan dari client >> pesan yang dikirim
Pesan dari client >> test doang
□
```

### Program Client [pyCChat.py]

```
#!/usr/bin/env python

from socket import *
import sys

class TCPCClient:
    def __init__(self):
        if len(sys.argv) != 3:
            print 'Penggunaan: ' + sys.argv[0] + ' [ip_server] [port]'
            sys.exit(1)
        else:
            self.HOST = sys.argv[1]
            self.PORT = sys.argv[2]

    def Create(self):
        try:
            self.sockTCP = socket(AF_INET,SOCK_STREAM) # Buat socket
            self.sockTCP.connect((self.HOST, int(self.PORT)))
```

```
except:
    print 'Socket error [ip_server dan port harus valid]'
    sys.exit()
else:
    print 'Client Message'
    print '-----'
    print 'Terhubung ke server %s' % self.HOST

def Send(self):
    try:
        while 1:
            pesan = raw_input('Pesan : ') # Input pesan
            self.sockTCP.send(pesan)      # Kirim pesan ke server
            data = self.sockTCP.recv(1024) # Terima pesan dari
server
            print data                    # Cetak pesan dari
server
    except:
        self.__del__()

def Run(self):
    self.Create()
    self.Send()

def __del__(self):
    self.sockTCP.close() # Tutup koneksi

if __name__ == '__main__':
    msg = TCPClient().Run()
```

### Penjelasan Program [pyCChat.py]

Program diatas merupakan program client chatting yang digunakan untuk melakukan koneksi ke server pySChat.py. Client akan melakukan koneksi ke server dan setelah terkoneksi client dapat mengirimkan pesan ke pada server dan server akan memberikan respon kepada client.

### Output Program [pyCChat.py]



```
drslump@darkstar:~/project/chat$ ./pyCChat.py 127.0.0.1 2000
Client Message
-----
Terhubung ke server 127.0.0.1
Pesan : pesan yang dikirim
[pesan yang dikirim] sudah diterima server.
Pesan : test doang
[test doang] sudah diterima server.
Pesan : █
```

### 3. Time server dengan socket Internet menggunakan UDP

#### Program Time Server [time-srv-udp.py]

```
#!/usr/bin/env python

from socket import *
import sys
import time

class ServerWaktu:
    def __init__(self):
        if len(sys.argv) != 3:
            print "penggunaan: " + sys.argv[0] + " <ip_address>
<port>"
            sys.exit(1)
        else:
            self.HOST = sys.argv[1]
            self.PORT = int(sys.argv[2])
            self.tHostPort = (self.HOST, self.PORT)

    def buatSocket(self):
        try:
            # Buat socket UDP
            self.tsock = socket(AF_INET, SOCK_DGRAM)
            print "Mengikatkan IP dengan port " + str(self.tHostPort)
            self.tsock.bind(self.tHostPort)
        except:
            print "Gagal buat socket"
            sys.exit()
```

```
def tungguKoneksi(self):
    print "Menunggu koneksi (ctrl-c keluar)..."
    while 1:
        try:
            data, client = self.tsock.recvfrom(0) # Terima data
            print "Ada permintaan dari client " + str(client)

            wkt = "Waktu server adalah " + time.ctime(time.time())
            self.tsock.sendto(wkt, client) # Kirim waktu server ke
client
        except:
            break

def tutupKoneksi(self):
    try:
        print "Tutup koneksi..."
        self.tsock.close()
    except:
        print "Gagal menerima data"

def __del__(self):
    self.tutupKoneksi()

if __name__ == "__main__":
    srv = ServerWaktu()
    srv.buatSocket()
    srv.tungguKoneksi()
```

### **Penjelasan Program [time-srv-udp.py]**

Program ini merupakan program yang berfungsi sebagai time server. Program ini menggunakan socket Internet dengan menggunakan protocol UDP. Cara kerja program ini adalah setiap ada koneksi/request dari client maka server akan mengirimkan waktu sistem yang menjalankan program server tersebut kepada client.

### Output Program [time-srv-udp.py]



```
Akses Root
brenkzek@brenkzek:/project$ ./time-srv-udp.py 127.0.0.1 2000
Mengikatkan IP dengan port ('127.0.0.1', 2000)
Menunggu koneksi (ctrl-c keluar)...
Ada permintaan dari client ('127.0.0.1', 1026)
█
```

### Program Time Client [time-cli-udp.py]

```
#!/usr/bin/env python

from socket import *
import sys

class ClientWaktu:
    def __init__(self):
        if len(sys.argv) != 3:
            print "penggunaan: " + sys.argv[0] + " <ip_address>
<port>"
            sys.exit(1)
        else:
            self.HOST = sys.argv[1]
            self.PORT = int(sys.argv[2])
            self.tHostPort = (self.HOST, self.PORT)

    def buatSocket(self):
        try:
            self.tsock = socket(AF_INET, SOCK_DGRAM)
            print "Hubungi server waktu di " + str(self.tHostPort)
            self.tsock.connect(self.tHostPort) # Hubungi server
            self.tsock.sendto("", self.tHostPort) # Kirim string
kosong
        except:
            print "Gagal buat socket"

    def terimaData(self):
        data, client = self.tsock.recvfrom(100) # Ambil data waktu
```

```
dari
    print data                                     # server

    def tutupKoneksi(self):
        print "Tutup koneksi..."
        self.tsock.close()

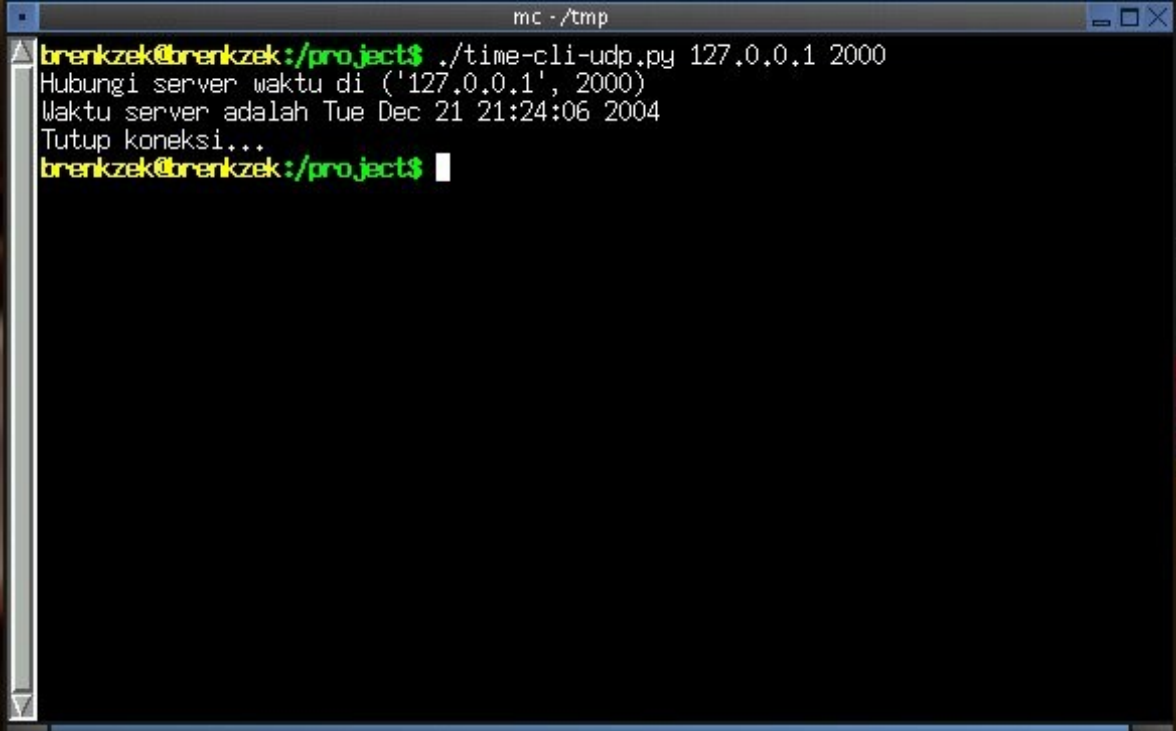
    def __del__(self):
        self.tutupKoneksi()

if __name__ == "__main__" :
    srv = ClientWaktu()
    srv.buatSocket()
    srv.terimaData()
```

### Penjelasan Program [time-cli-udp.py]

Program diatas merupakan program time client yang berfungsi untuk meminta waktu server. Program diatas menggunakan socket Internet dengan menggunakan protocol UDP.

### Output Program [time-cli-udp.py]



```
mc - /tmp
brenkzek@brenkzek:/project$ ./time-cli-udp.py 127.0.0.1 2000
Hubungi server waktu di ('127.0.0.1', 2000)
Waktu server adalah Tue Dec 21 21:24:06 2004
Tutup koneksi...
brenkzek@brenkzek:/project$
```

#### 4. Program Chatting dengan socket UNIX menggunakan protocol TCP

##### Program Server [pyUnixS.py]

```
#!/usr/bin/env python

from socket import *
import sys
import os

class TCPServer:
    def __init__(self):
        self.path = '/tmp/sock' # Path untuk socket yang akan dibuat

    def Create(self):
        try:
            self.sockUnix = socket(AF_UNIX, SOCK_STREAM) # Buat
socket TCP
            self.sockUnix.bind(self.path) # Binding addr dengan port
            self.sockUnix.listen(1) # Listening
        except:
            print 'Socket sudah digunakan...\n\rJika belum, hapus file
'+ \
            self.path
            sys.exit()
        else:
            print 'Server Message [tekan Ctrl-C untuk keluar]'
            print '-----'

    def Accept(self):
        koneksi, alamat = self.sockUnix.accept() # Terima koneksi
        print 'Ada koneksi...'
        while 1:
            try:
                data = koneksi.recv(1024) # Terima data
                print 'Pesan dari client >> ' + str(data)
                if not data: break
                if len(data) > 1: # Kirim respon ke client
                    koneksi.send('[' + str(data)+ '] sudah diterima
server.')
```

### Penjelasan Program [UnixS.py]

Program diatas merupakan program chatting. Sebenarnya program diatas sama dengan program 2. Bedanya kalau pada program 2 menggunakan socket Internet sedangkan pada program diatas menggunakan socket Unix. Seperti yang telah dijelaskan pada bagian awal bahwa pada socket Unix socket dibuat dalam bentuk file socket disini digunakan file `/tmp/sock`. Jika pada saat dijalankan terjadi error coba Anda hapus terlebih dahulu file socket yang ada di `/tmp/sock`. Cara kerja programnya adalah server akan menunggu koneksi dari client dan jika ada client yang menghubungi server maka server akan menerima koneksi tersebut kemudian client dapat mengirimkan pesan ke server. Setiap pesan yang dikirimkan ke server maka server akan memberi respon balik ke client. Pada saat program keluar maka program akan menghapus file socket yang ada di `/tmp/sock`.

### Output Program [UnixS.py]



```
mc - ~/project
drslump@darkstar:~/project$ ./UnixS.py
Server Message [tekan Ctrl-C untuk keluar]
-----
Ada koneksi...
Pesan dari client >> coba neeh
Pesan dari client >> ini menggunakan socket unix
□
```

## Program Client [UnixC.py]

```
#!/usr/bin/env python

from socket import *
import sys

class TCPCClient:
    def __init__(self):
        self.path = '/tmp/sock' # Path untuk melakukan koneksi ke
socket

    def Create(self):
        try:
            self.sockUnix = socket(AF_UNIX, SOCK_STREAM) # Buat socket
UNIX

            self.sockUnix.connect(self.path)
        except:
            print 'Socket error [ip_server dan port harus valid]'
            sys.exit()
        else:
            print 'Client Message'
            print '-----'

    def Send(self):
        try:
            while 1:
                pesan = raw_input('Pesan : ') # Meminta input pesan
                self.sockUnix.send(pesan) # Mengirim pesan ke client
                data = self.sockUnix.recv(1024) # Menerima respon dari
server

                print data
        except:
            self.__del__()

    def Run(self):
        self.Create()
        self.Send()

    def __del__(self):
        self.sockUnix.close()

if __name__ == '__main__':
    msg = TCPCClient().Run()
```

## Output Program [UnixC.py]



```
mc - ~/project
dhs lump@darkstar:~/project$ ./UnixC.py
Client Message
-----
Pesan : coba neeh
[coba neeh] sudah diterima server.
Pesan : ini menggunakan socket unix
[ini menggunakan socket unix] sudah diterima server.
Pesan : █
```

## Penjelasan Program [UnixC.py]

Program diatas merupakan program chatting client. Setiap client mengirimkan pesan ke server maka server akan mengirimkan respon balik ke client.

## 5. Program FTP Server dan Client Sederhana

### Program FTP Server [pyFTPd.py]

```
#!/usr/bin/env python

from socket import *
import commands
import os
import sys
import time
import string

lstCmd = ['dir', 'ls', 'exit', 'bye', 'quit', 'get', 'mget', 'put',
          'mput', 'rm', 'delete', 'mv', 'rename', 'cd', 'pwd', 'chmod',
          'cp', 'copy', 'rmdir', 'mkdir', 'close', 'disconnect']
hostIP = '127.0.0.1'
hostPort = 1111
filFlag = '*file*'
getFlag = 'get'
```

```
class CMD:
    def __init__(self):
        self.byeFlag = '*bye*'

    def checkCmd(self, cmd):
        ret = ''
        cmd = cmd.strip().split()
        cmd[0] = cmd[0].lower()

        if cmd[0] in lstCmd:
            if cmd[0] in ['ls', 'dir']:
                if len(cmd) == 1:
                    cmdS = 'ls -al'
                else:
                    cmdS = 'ls' + ' ' + cmd[1]
                ret = commands.getoutput(cmdS)

            elif cmd[0] in ['rm', 'delete']:
                if len(cmd) == 2:
                    ret = commands.getoutput('rm ' + cmd[1])
                    if ret == '':
                        ret = 'File ' + cmd[1] + ' telah di hapus.'
                else:
                    ret = 'penggunaan: rm|delete [file].'

            elif cmd[0] in ['rmdir']:
                if len(cmd) == 2:
                    ret = commands.getoutput('rm -rf ' + cmd[1])
                    if ret == '':
                        ret = 'Direktori ' + cmd[1] + ' telah di hapus.'
                else:
                    ret = 'rmdir [direktori].'

            elif cmd[0] in ['mkdir']:
                if len(cmd) == 2:
                    ret = commands.getoutput('mkdir ' + cmd[1])
                    if ret == '':
                        ret = 'Direktori ' + cmd[1] + ' telah di buat.'
                else:
                    ret = 'penggunaan: mkdir [direktori].'

            elif cmd[0] in ['mv', 'rename']:
                if len(cmd) == 3:
                    ret = commands.getoutput('mv ' + cmd[1] + ' ' +
cmd[2])
                else:
                    ret = 'penggunaan: mv|rename [file_lama]
[file_baru].'

            elif cmd[0] in ['cp', 'copy']:
                if len(cmd) == 3:
                    ret = commands.getoutput('cp ' + cmd[1] + ' ' +
cmd[2])
                else:
                    ret = 'penggunaan: cp|copy [file_sumber]
[tujuan].'

            elif cmd[0] in ['chmod']:
                if len(cmd) == 3:
```

```
        ret=commands.getoutput('chmod ' + cmd[1] + ' ' +
cmd[2])
        if ret == '':
            ret = 'Hak akses ' + cmd[1] + ' telah di ubah.'
        else:
            ret = 'penggunaan: chmod [mode] [file].'
```

```
elif cmd[0] in ['cd']:
    if len(cmd) == 2:
        try:
            os.chdir(cmd[1])
        except:
            ret = 'Direktori tidak ada.'
        else:
            ret = 'Direktori sekarang ' + os.getcwd()
    else:
        ret = 'penggunaan: cd [direktori]'
```

```
elif cmd[0] in ['pwd']:
    ret = 'Direktori sekarang ' + commands.getoutput('pwd')
```

```
elif cmd[0] in ['bye','exit','quit','close','disconnect']:
    ret = self.byeFlag
```

```
return ret
```

```
class CLI:
    def __init__(self):
        self.cmd = CMD()
        self.childLst = []

    def updatePid(self, pids):
        while self.childLst:
            pid, status = os.waitpid(0, os.WNOHANG)
            if not pid : break
            pids.remove(pid)

    def sendFile(self, sock, file):
        sock.send(filFlag)
        user = os.environ['USER']
        command = filFlag
        size = os.stat(file)[6]
        try:
            f = open(file,'r')
        except:
            ret = 0
        else:
            pos = 0
            while 1:
                if pos == 0:
                    buffer = f.read(5000000-282)
                    if not buffer: break
                    count = sock.send(command + ':' + \
string.rjust(os.path.basename(file),214) + ':' + \
string.rjust(str(size).strip(),30) + ':' + \
string.rjust(str(user).strip(),30) + \
buffer)
                    pos = 1
                else:
```

```
        buffer = f.read(5000000)
        if not buffer: break
        count = sock.send(buffer)

    ret = 1
    return ret

def recvFile(self, sock) :
    pjpg = 0
    msg1 = sock.recv(283).split(':')
    flag = msg1[0].strip()
    namafile = msg1[1].strip()
    total = msg1[2].strip()
    user = msg1[3].strip()
    file = namafile
    if flag == filFlag:
        try:
            f = open(file, 'w')
        except:
            ret = 0
        else:
            try:
                while 1:
                    leftToRead = int(total) - pjpg
                    if not leftToRead: break
                    msg = sock.recv(5000000)
                    pjpg = pjpg + len(msg)
                    f.write(msg)
                f.close()
            except:
                os.remove(file)
                ret = 0
            else:
                ret = 1
    ret = 1
    return ret

def handler(self, sock) :
    sock.send('\n\rSelamat datang di DrSlump FTP Server\n\r')
    while 1:
        data = sock.recv(1024)
        if not data: break
        if data[:len(filFlag)] == filFlag:
            ret = self.recvFile(sock)
            if ret == 1:
                ret = 'File telah di terima'
            else:
                ret = 'File gagal di terima'
            sock.send(ret)
        elif data[:len(getFlag)] == getFlag:
            cmd = data.strip().split()
            cmd[0] = cmd[0].lower()
            self.sendFile(sock, cmd[1])
        else:
            ret = self.cmd.checkCmd(data)
            if ret == self.cmd.byeFlag:
                sock.send('Koneksi di tutup...\n\r')
                sock.close()
                break
            sock.send(str(ret) + '\n\r')
```

```
def runCmd(self):
    print '\n\r + + + Dr_slump FTP Server + + +\n\r'
    try:
        print 'Membuat socket...'
        sockCmd = socket(AF_INET, SOCK_STREAM)
    except:
        print 'Gagal buat socket !'
        sys.exit()
    else:
        sockCmd.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
        try:
            print 'Mengikatkan socket ke ' + hostIP + ' port'+
str(hostPort)
            sockCmd.bind((hostIP, hostPort))
        except:
            print 'Gagal mengikatkan socket !'
            sys.exit()
        else:
            print 'Mendengarkan koneksi...\n\r'
            sockCmd.listen(5)
            while 1:
                koneksi, alamat = sockCmd.accept()
                self.updatePid(self.childLst)
                pid = os.fork()
                if pid: #parent
                    koneksi.close()
                    self.childLst.append(pid)

                if not pid: #child
                    sockCmd.close()
                    self.handler(koneksi)
                    os._exit(0)

if __name__ == "__main__":
    cli = CLI()
    cli.runCmd()
```

### Output Program [pyFTPd.py]



```
mc - ~/project/ftp
drslump@darkstar:~/project/ftp$ ./pyFTPd.py

+ + + Dr_slump FTP Server + + +

Membuat socket...
Mengikatkan socket ke 127.0.0.1 port 1111
Mendengarkan koneksi...

[]
```

### Program Client [pyFTP.py]

```
#!/usr/bin/env python

from socket import *
import os
import sys
import time
import string

lstCmd = ['dir', 'ls', 'exit', 'bye', 'quit', 'clear', 'cls', 'get', 'mget',
          'put', 'mput', 'rm', 'delete', 'mv', 'rename', 'cd', 'pwd', 'chmod',
          'cp', 'copy', '?', 'help', 'rmdir', 'mkdir', '!', 'connect', 'open',
          'close', 'disconnect']

defPort = 1111
filFlag = '*file*'

class IO:
    def connect(self, host, port):
        try:
            print 'Membuat socket...'
            self.sockIO = socket(AF_INET, SOCK_STREAM)
        except:
            print 'Gagal membuat socket !'
            ret = 0
        else:
            try:
                print 'Koneksi ke ' + host + ' port ' + str(port)
```

```
        self.sockIO.connect((host, port))
    except:
        print 'Koneksi gagal!\n\r'
        ret = 0
    else:
        print 'Koneksi sukses...\n\r'
        data = self.sockIO.recv(1024)
        print data
        ret = self.sockIO
    return ret

def sendFile(self, sock, file) :
    sock.send(filFlag)
    user = os.environ['USER']
    command = filFlag
    size = os.stat(file)[6]
    f = open(file, 'r')
    pos = 0
    while 1:
        if pos == 0:
            buffer = f.read(5000000-282)
            if not buffer: break
            count = sock.send(command + ':' + \
                string.rjust(os.path.basename(file), 214) + ':' + \
                string.rjust(str(size).strip(), 30) + ':' + \
                string.rjust(str(user).strip(), 30) + \
                buffer)
            pos = 1
        else:
            buffer = f.read(5000000)
            if not buffer: break
            count = sock.send(buffer)

def recvFile(self, sock) :
    pjpg      = 0
    msg1      = sock.recv(283).split(':')
    flag      = msg1[0].strip()
    namafile  = msg1[1].strip()
    total     = msg1[2].strip()
    user      = msg1[3].strip()
    file      = namafile

    if flag == filFlag:
        try:
            f = open(file, 'w')
        except:
            ret = 0
            print 'Tidak dapat menyimpan file'
            sys.exit()
        else:
            try:
                while 1:
                    leftToRead = int(total) - pjpg
                    if not leftToRead: break
                    msg = sock.recv(5000000)
                    pjpg = pjpg + len(msg)
                    f.write(msg)
                    os.system('echo -n !')
```

```

        f.close()
    except:
        os.remove(file)
        ret = 0
    else:
        ret = 1

    def close(self):
        self.sockIO.close()

class CMD:
    def __init__(self):
        self.getFlag = '*get*'
        self.putFlag = '*put*'
        self.IO      = IO()
        self.isConnected = 0

    def checkCmd(self, cmd):
        ret = 0
        cmd0= cmd
        cmd = cmd.strip().split()
        cmd[0] = cmd[0].lower()

        if cmd[0] or cmd[0][0] in lstCmd:
            if cmd[0] in ['?', 'help']:
                print '\n\rDaftar perintah yang digunakan: \n\r\n\r' +
\
                '?                disconnect                mv
[file_lama] [file_baru]\n' +\
                'bye                exit                open
[host]\n' +\
                'cd [direktori]                get [file]                put
[file]\n' +\
                'chmod [mode] [file] help                pwd\n' +\
                'clear                ls [direktori|file] rename
[file_lama] [file_baru]\n' +\
                'cls                rm [file]                rmdir
[direktori]\n' +\
                'connect [host]                mget [files]
quit\n' +\
                'delete [file]                mkdir [direktori]                !
[perintah_lokal]\n' +\
                'dir [direktori|file]                mput [files]\n\r'

            elif cmd[0] in ['connect', 'open']:
                if not self.isConnected:
                    if len(cmd) == 2:
                        self.Sock = self.IO.connect(cmd[1], defPort)
                        if self.Sock <> 0:
                            self.isConnected = 1
                        else:
                            self.isConnected = 0
                    else:
                        print 'penggunaan: connect|open [host]'
                else:
                    print 'Tutup koneksi dulu...'


            elif cmd[0] in ['clear', 'cls']:
                os.system('clear')

```

```
elif cmd[0] in ['put']:  
    if self.isConnected:  
        if os.path.isfile(cmd[1]):  
            ret = 1  
            self.IO.sendFile(self.Sock, cmd[1])  
        else:  
            print 'Gagal membaca file'  
    else:  
        print 'penggunaan: put [file]'  
  
elif cmd[0] in ['bye', 'exit', 'quit', 'close', 'disconnect']:  
    if self.isConnected:  
        self.Sock.send(cmd[0])  
        self.isConnected = 0  
        print self.Sock.recv(100)  
        ret = 1  
    else:  
        print 'Goodbye...\n\r'  
        sys.exit()  
  
elif cmd[0][0] == '!':  
    if len(cmd[0]) == 2:  
        if cmd[0][1:] == 'cd':  
            print os.chdir(cmd[2])  
        else:  
            print os.system(cmd[1:])  
    else:  
        print os.system(cmd[1:])  
  
else:  
    try:  
        self.Sock.send(cmd[0])  
        ret = 1  
    except:  
        print 'Tidak terkoneksi !'  
  
else:  
    print 'Perintah tidak dikenal.'  
  
return ret  
  
def runCmd(self):  
    print '\n\r+ + + DrSlump FTP Client + + +\n\r'  
    while 1:  
        cmd = raw_input('ftp> ')  
        if len(cmd.strip()) > 0:  
            ret = self.checkCmd(cmd)  
            if self.isConnected and ret == 1:  
                data = self.Sock.recv(500000)  
                if data[:len(filFlag)] == filFlag:  
                    ret = self.IO.recvFile(self.Sock)  
                    if ret == 0:  
                        print '\n\rFile gagal di download !'  
                    else:  
                        print '\n\rFile berhasil di download...'  
            else:  
                print data
```

```
if __name__ == "__main__":  
    cmd = CMD()  
    cmd.runCmd()
```

### Output Program [pyFTP.py]



```
drslump@darkstar:~/project/ftp$ ./pyFTP.py  
  
+ + + DrSlump FTP Client + + +  
  
ftp> connect 127.0.0.1  
Membuat socket...  
Koneksi ke 127.0.0.1 port 1111  
Koneksi sukses...  
  
Selamat datang di DrSlump FTP Server  
  
ftp> ls  
total 32  
drwxr-xr-x  2 drslump  users      4096 Sep 12 13:17 .  
drwxr-xr-x  4 drslump  users      4096 Dec 15 21:36 ..  
-rwxr-xr-x  1 drslump  users      4685 Sep 12 13:16 pyFTP.py  
-rwxr-xr-x  1 drslump  users      5589 Dec 16 09:51 pyFTPd.py  
-rwxr-xr-x  1 drslump  users      4266 Sep  8 11:53 pyFTPd2.py  
  
ftp> █
```

### Penjelasan Program

Program FTP Server dan Client diatas merupakan program FTP sederhana. Program diatas dapat mengirimkan dan menerima file, melakukan operasi file/direktori seperti cd dan mkdir dan lain-lain. Program server bertipe *concurrent server* yang dapat melayani banyak client dengan menggunakan threading.

## Penutup

Pembuatan aplikasi jaringan dengan Python sebenarnya tidak terlalu sulit, karena Python telah menyediakan modul-modul siap pakai yang memudahkan pemrogram untuk membuat program. Sebenarnya selain menggunakan modul socket banyak modul-modul lain yang dapat digunakan untuk membuat aplikasi jaringan seperti `telnetlib`, `ftplib`, `httpplib`, `serversocket` dan lain-lain.

Penulis menyadari tulisan ini jauh dari sempurna dan bisa dikatakan sangat sederhana, tetapi Penulis berharap dengan tulisan ini dapat memberikan informasi yang berguna bagi Anda. Dan jika masih ada kekurangan mohon harap dimaklumi. Saran, kritik dan koreksi sangat diharapkan. ;)

## Referensi

Noprianto, **Python dan Pemrograman Linux**. Andi Offset, Yogyakarta, 2002.

Rossum, Guido van., **Python Documentation**. Release2.2.2, [www.python.org/doc](http://www.python.org/doc), 2002.

Stevens, W.Richard., **UNIX Network Programming Networking APIs**. Volume 1 Second Edition, Prentice Hall PTR, US, 1998.

Sebastian V. Tponut, **Python Network Programming**, Technical University Timisoara  
Version 0.00, 16. July 2001

## Biografi Penulis



**Rikih Gunawan**. Tinggal di Serpong, menyelesaikan S1 di universitas Gunadarma, jurusan Teknik Informatika (2001-2004).