简易聊天室程序说明文档

姓 名：余锋伟

学 号：11091222

日 期：2014.5.23

目录

1 引言

1.1 编写目的

本文档介绍了网络应用编程实验要求实现的通信程序，分别从需求、设计、编码等角度对该程序做了详细的介绍。

本文档的对象是关注 Socket 通信程序实现，有基本的 Python 编程语言能力的读者。可以用过阅读本文档，了解程序实现的技术细节，对 Python 实现多线程 Socket 程序有一个直观的认识。

1.2 背景

该通信程序是作为网络实验编程实践部分被提出的，由参与网络实验学习的学生独立编写和测试。作为一个网络通信程序，其运行环境可以使单机的，也可以是多机的。

按照网络实验课程对该通信程序的要求，需要使用的 Socket 和多线程两项主要技术。完成的通信程序可以运行在学生自选的操作系统上。

1.3 定义

下面罗列了程序中使用的一些主要术语的定义：

- socket ：通常也称作“套接字”，用于描述 IP 地址和端口，是一个通信链的句柄。应用程序通常通过“套接字”向网络发出请求或者应答网络请求。在连接成功时，应用程序两端都会产生一个 Socket 实例，操作这个实例，完成所需的会话。
- 线程：进程中的一个实体，是被系统独立调度和分派的基本单位。线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。
- 多线程：线程是程序中一个单一的顺序控制流程，在单个程序中同时运行多个线程完成不同的工作，称为多线程；一个线程可以创建和撤消另一个线程，同一进

程中的多个线程之间可以并发执行。

- PyQt：**PyQt** 是一个创建 GUI 应用程序的工具包。它是 Python 编程语言和 Qt 库的成功融合。Qt 库是目前最强大的库之一。PyQt 是由 **Phil Thompson** 开发。 **PyQt** 实现了一个 Python 模块集。它有超过 300 类，将近 6000 个函数和方法。它是一个多平台的工具包，可以运行在所有主要操作系统上，包括 UNIX，Windows 和 Mac。 **PyQt** 采用双许可证，开发人员可以选择 GPL 和商业许可。

2 软件需求说明

2.1 任务概述

2.1.1 目标

按照网络实验课程对该通信程序的要求，需要使用的 Socket 和多线程两项主要技术。完成的通信程序可以运行在学生自选的操作系统上。

本软件是一个可独立运行的通信程序，而不是一个更大系统的组成部分。其实现的功能是使用 Socket 和多线程技术实现一个聊天程序，支持多人同时聊天（聊天室功能）。

由于该程序是使用 Python 语言开发的，其运行时需要有 Python 解释环境和 PyQt4 库的支持，本程序开发过程中使用的是 Python2.7。

2.1.2 用户的特点

该软件的用户即是参与到聊天群组当中的个人。观察目前较流行的商业聊天软件，如 QQ、MSN 等可以发现，参与到聊天过程中的人具有不同的教育水平和技术专长，这就要求软件的使用界面应尽量简单，方便各种层次的用户使用。

该软件作为网络实验课程的实践部分，可以预见，其今后的应用范围和频度都比较小，基本上局限于同学之间内部使用，没有广泛的用户群，软件内部的容错、均衡机制可以暂不考虑。

2.1.3 假定和约束

进行本软件开发工作的假定和约束主要有：

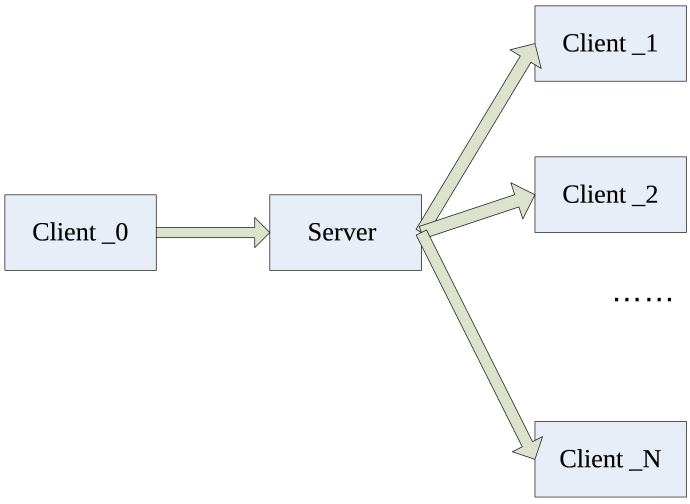
- 相关程序和文档由个人独立完成；
- 程序开发过程中需要使用到 Socket 和多线程技术；
- 实现基本的多人聊天功能；

2.2 需求规定

2.2.1 对功能的规定

本软件需要提供的功能为对多人聊天的支持。

下图展示的是多人聊天功能。系统中的每一个用户都可以向同时在线的其他用户发送消息。



2.2.3 输入输出要求

该软件完成的功能主体是网络聊天。其输入数据主要是是系统中的用户输入的文本信息；输出是把一个用户输入的文本信息传送，并输出到另一个用户的界面上。

输入采用 unicode 编码，支持中文显示。由于每次发送的聊天信息只有用户名+消息，

所以一次信息占用一行。

2.3 运行环境规定

2.3.1 设备

该程序使用目前常规的 PC 机器开发，机器的主要配置有：

- 处理器: Intel Core(TM)2 Duo Processor
- 主板: LENOVO LENOVO
- 芯片组: Intel P35/G33/G31/P31 Express DRAM Controller
- 内存: 2096300 KB
- 主硬盘: Hitachi HDP725025GLA380
- 网络适配器: Marvell Yukon 88E8056 PCI-E Gigabit Ethernet Controller

由于该程序完成的功能主要是网络通信，没有其他特殊应用，所以目前常见的 PC 机器都可以运行。

2.3.2 支持软件

该程序使用 Python 语言开发，运行时需要有 Python 解释机和 PyQt4 包的支持。

由 Python 语言以及 Qt 的跨平台特性，该程序既可以运行在 Windows 操作系统上，也可以运行在 Linux 操作系统上。

3 详细设计说明

3.1 程序系统的结构

按照事先的功能，可将程序可划分为两个部分：一部分是服务器，它负责监听端口，为每个连入的用户提供数据转发服务；另一部分是客户端，负责和服务器建立连接，监听

服务器发送过来的数据，并提交给用图形界面显示

服务器端采用了并发服务的模式，它是一种高效的服务器运行模式，允许服务器同时与多个客户端进行通信。其实现的方法是，服务器端一直处于端口监听状态，一旦有新的客户连接请求就创建一个线程，负责与这个客户进行通信，当这个客户端中断连接或者发生错误后结束相应的服务线程。

客户端同样使用了线程机制。用户将看到一个主界面，主界面展示当前的链接状态，用户看不到的是，该窗口在创建完成后会建立一个子进程，负责监听服务器端发来的数据并把这些数据提交给相应的窗口展示出来。

3.2 服务器设计说明

3.2.1 程序描述

通过本程序启动服务器，将创建一个服务器端口监听程序，负责为每一个接入系统的合法用户创建一个服务线程。程序在退出时，将会杀掉后台的监听进程以及存在的用户服务线程。

3.2.2 功能

简单执行 Python server.py，即运行服务器。服务器讲在 8001 端口监听链接请求。对于每个链接请求新建一个线程处理消息。

3.2.3 输入/输出

程序没有输入，对于输出，每次新建一个链接时都会显示客户端的 ip 和端口号。每次客户端发送消息时，都在控制台上显示发送的姓名和消息。

3.3 客户服务程序设计说明

3.3.1 程序描述

客户服务程序是由服务器端口监听程序创建的线程，它负责为每个接入系统的用户提供数据转发的服务。这部分程序是服务器端程序的核心部分。

3.3.2 流程逻辑

不断监听客户程序发送的数据，如果为“QUIT”，则删除该套接字，退出线程。否则转发该程序到所有在线的用户。

```
p[forwil@localhost socket]$ python server.py
ip: 127.0.0.1:43170 is connected!
foo :Hello!
QUIT
```

3.4 客户主界面设计说明

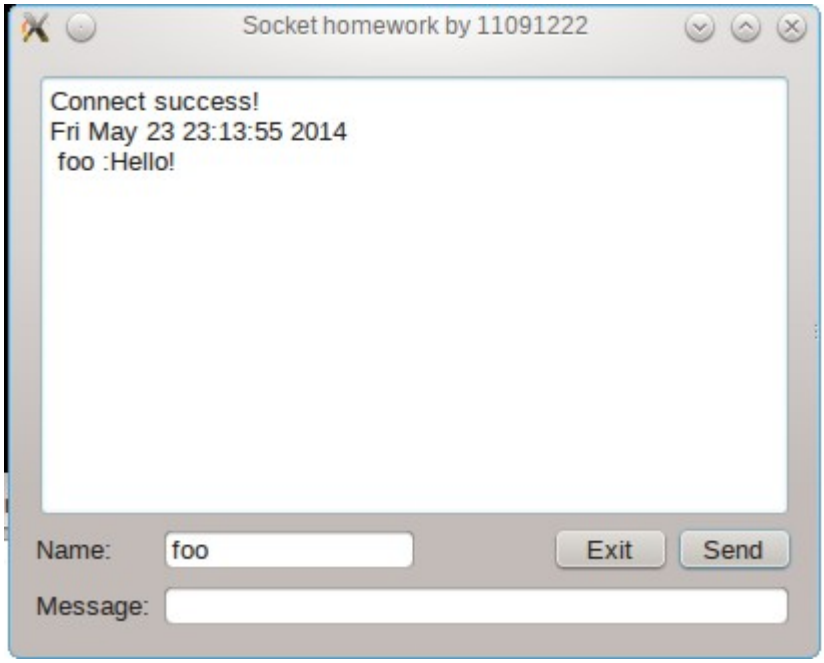
3.4.1 程序描述

客户主界面是为当前用户提供了 2 个输入信息窗口，和 2 个功能按钮。

3.4.2 功能

该程序提供了四个功能：

- 连接服务器
- 更改用户名
- 断开服务器
- 发送信息



用户主界面

3.4.3 输入/输出

输入为用户点击的按钮对应的事件号。

输出为当前的聊天信息，以及变动的连接状态。

4 编码实现

Client.py

```
#coding =utf8
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

import thread
import socket
import time
```

```
from PyQt4 import QtGui
import PyQt4.Qt as Qt

app = QtGui.QApplication(sys.argv)

widget = QtGui.QWidget()
widget.resize(400, 300)
widget.setWindowTitle('Socket homework by 11091222')

layout = QtGui.QGridLayout(widget)

text = QtGui.QTextBrowser()
lmessage = QtGui.QLabel("Message:")
message = QtGui.QLineEdit()
lname = QtGui.QLabel("Name:")
name = QtGui.QLineEdit()
name.setText("foo")
send = QtGui.QPushButton("Send")
conn = QtGui.QPushButton("Conn")
send.setDisabled(True)

layout.addWidget(text,0,0,6,6)
layout.addWidget(lname,6,0)
layout.addWidget(name,6,1,1,2)
layout.addWidget(send,6,5,1,1)
layout.addWidget(conn,6,4,1,1)
layout.addWidget(lmessage,7,0,1,1)
layout.addWidget(message,7,1,1,5)

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

port = 8001
host = 'localhost'

def sendfunc(sender):
        if len(message.text())<=0:
                return
        buf = name.text() + " :"+message.text()
        message.clear()
        sock.send(unicode(buf))
        return

def connfunc(sender):
```

```
        global sock
        if send.isEnabled() == False:
                try:
                        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                        sock.connect((host,port))
                        send.setDisabled(False)
                        text.append("Connect success!")
                        conn.setText("Exit")
                except:
                        text.append("Connect fault!please check your network")
        else:
                send.setDisabled(True)
                sock.send("QUIT")
                sock.close()
                text.append("Connect close,Good-bye!")
                conn.setText("Conn")


def recv():
        while True:
                try:
                        ss = unicode(sock.recv(1024))
                        print ss
                        if len(ss)>1:
                                text.append("%s \n %s" % (time.asctime(),ss))
                except:
                        pass
        thread.exit_thread()

send.mousePressEvent = sendfunc
conn.mousePressEvent = connfunc

widget.show()
thread.start_new_thread(recv,())
app.exec_()
sys.exit(sock.close())
```

Server.py

```
#coding= utf8
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
```

```
import socket
import thread

sockets = []

def deal(connection):
        while True:
                try:
                        buf = connection.recv(1024)
                        print buf
                        if buf == "QUIT":
                                sockets.remove(connection)
                                thread.exit_thread()
                        for i in sockets:
                                i.send(buf)
                except :
                        break


if __name__ == '__main__':
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.bind(('localhost', 8001))
        sock.listen(5)
        while True:
                connection,address = sock.accept()
                print "ip: %s:%d is connected!" % (address[0],address[1])
                thread.start_new_thread(deal,(connection,))
                sockets.append(connection)
```

5 总结

通过编写这个聊天程序，使我基本掌握了使用 Python 语言进行 C/S 模式程序开发的方法。

通过实际编程使用 Socket 通信和多线程同步机制，使我对这两种技术的实现方式也有了较清晰的认识。