

**计算机网络课程设计报告**

**系 （院）：**  计算机科学学院

**专业班级：** 软工实11401

**姓 名：** 周逸飞

**学 号：** 201406942

**指导教师：** 宦成林

**设计时间：** 2017.6.12 - 2017.6.23

**设计地点：**  4教5楼机房

目录

一．需求分析 3

1. 引言 3

1.1 背景 3

1.2 参考资料 3

1.3 假定和约束 4

1.4 开发环境 4

1.5 用户的特点 4

2. 任务概述 4

2.1 系统总体流程 4

2.2 需求分析 5

二．概要设计 5

三．详细设计 6

1. 客户端 6

1.1 功能模块 6

1.2 运行流程 8

2. 服务端 8

2.1 功能模块 8

2.2 运行流程 11

四．运行结果 11

五．总结 13

六．附录 13

# 需求分析

## 1. 引言

### 1.1 背景

计算机网络课程设计，需要学生们通过使用c/c++语言进行socket通信的编程实践。以此让学生们能够更进一步的了解TCP、UDP在真实环境下的运行机制。介于以上原因，我选择了《基于UDP的简易聊天机器人》这个题目进行开发。

UDP协议全称是用户数据报协议，在网络中它与TCP协议一样用于处理数据包，是一种无连接的协议。在OSI模型中，在第四层——传输层，处于IP协议的上一层。UDP有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用UDP协议。

建立网络通信连接至少要一对端口号(socket)。socket本质是编程接口(API)，对TCP/IP的封装，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口；HTTP是轿车，提供了封装或者显示数据的具体形式;Socket是发动机，提供了网络通信的能力。

Winsock2 SPI（Service Provider Interface）服务提供者接口建立在Windows开放系统架构WOSA（Windows Open System Architecture）之上，是Winsock系统组件提供的面向系统底层的编程接口。Winsock系统组件向上面向用户应用程序提供一个标准的API接口；向下在Winsock组件和Winsock服务提供者（比如TCP/IP协议栈）之间提供一个标准的SPI接口。各种服务提供者是Windows支持的DLL，挂靠在Winsock2 的Ws2\_32.dll模块下。对用户应用程序使用的Winsock2 API中定义的许多内部函数来说，这些服务提供者都提供了它们的对应的运作方式(例如API函数WSAConnect有相应的SPI函数WSPConnect)。多数情况下，一个应用程序在调用Winsock2 API函数时，Ws2\_32.dll会调用相应的Winsock2 SPI函数，利用特定的服务提供者执行所请求的服务。

### 1.2 参考资料

[**【windows socket+UDP服务器客户端】 - Jiangwei的专栏**](http://blog.csdn.net/lovecodeless/article/details/25388291)

### 1.3 假定和约束

开发期限：1-2周的时间，截止2017年6月23日星期五之前完成。

### 1.4 开发环境

操作系统：windows10家庭版

编辑软件：CodeBlocks

编程语言：C

编译器：MinGW（GCC）

### 1.5 用户的特点

看重系统功能的完善程度，系统能够按照要求稳定的运行。对界面的友好程度要求不高。

## 2. 任务概述

### 2.1 系统总体流程

第一步，运行服务端程序，服务端程序在启动时将机器人聊天文本robot.txt加载至内存当中。

1. 根据文件路径获取文件指针。
2. 判断是否到文件末尾，有则读取完毕，没有则继续下一步操作
3. 读取文件的第一行数据
4. 问题与答案通过‘\t’分隔开来，分别将经过分割的字符串保存至问题列表和答案列表当中
5. 返回（2）

第二步，运行客户端程序，进入等待输入状态。

第三步，输入你计划好的对话文字，按下回车键，将相应的信息发送至服务端。

第四步，服务端接收到对话请求，在聊天文本中搜寻与之相对应的响应信息，如果没有搜寻到，则返回‘sorry,I connot understand!’。

第五步，客户端接收到服务端的响应信息，并在控制台上进行输出。

第六步，返回至第二步。

### 2.2 需求分析

#### 2.2.1 客户端连接聊天机器人服务器

功能描述：客户端能够通过通过套接字和服务端程序进行通信。

#### 2.2.2 消息发送

功能描述：客户端发送消息给机器人服务器。并且客户端发送为纯文本的消息。

#### 2.2.3 消息接收

功能描述：客户端接收到机器人服务器发送给他的消息。

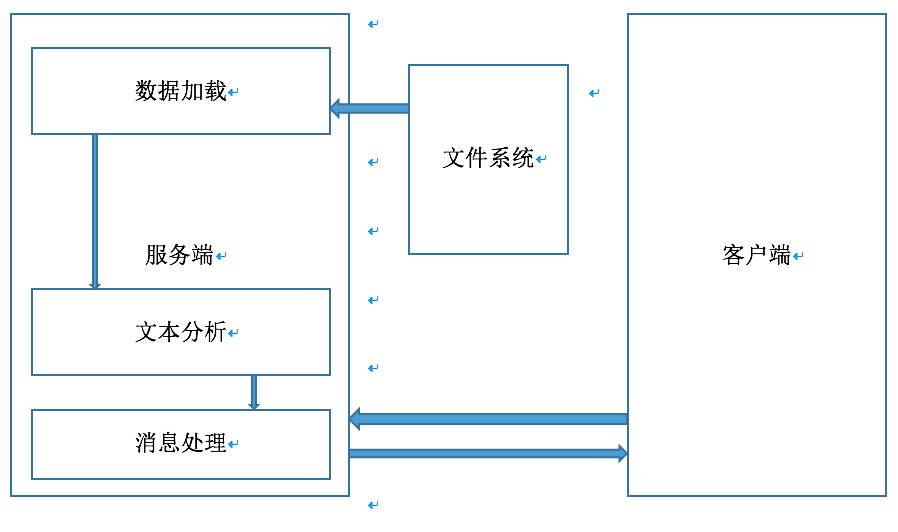
#### 2.2.4. 可以有多个客户端同时连接

功能描述：可以同时有多个客户端向服务端机器人发送请求，并能获得不同的响应。

#### 2.2.5. 智能回复功能

功能描述：根据用户发送的消息内容，稍微有点智能回复。

# 二．概要设计



程序结构框图

# 三．详细设计

## 1. 客户端

### 1.1 功能模块

#### 1.1.1 基于UDP协议的socket的通信

int socket(int af, int type, int protocol)

返回值：int

函数名称：socket

参数列表

int af：一个描述地址，目前仅支持AF\_INET格式，与就是说ARPA Internet地址格式。

int type：指定socket类型。新套接口的类型描述类型，如TCP（SOCK\_STREAM）和UDP（SOCK\_DGRAM）。常用的socket类型有，SOCK\_STREAM, SOCK\_DGRAM, SOCK\_PACKET, SOCK\_SEQPACKET等等。

int protocol：套接字所使用的协议。如果调用者不想指定，可使用0.常用的协议有，IPPRPTP\_TCP, IPPROTO\_UDP, IPPROTO\_STCP, IPPROTO\_TIPC，它们分别对应TCP传输协议、UDP传输协议、STCP传输协议、TIPC传输协议。

函数说明：用于根据制定的地址族，数据类型和协议来分配一个套接字的描述字及其所用的资源。

int sendto(socket s, const void \*msg, int len, unsigned int flags, const struct sockaddr \*to, int tolen)

返回值：int（成功则返回实际传出的字符数，失败返回-1）

函数名称：socket

参数列表

socket s：已经建好连线的socket

const void\* msg：指向想要连线的数据内容

unsigned int flags：一般设置为0

const struct sockaddr\* to：

int tolen：sockaddr的结果长度

函数说明：用来将数据由指定的socket传给对方主机。

ssize\_t recvfrom(int sockfd, void \*buf, size\_t len, unsigned int flags, struct sockaddr \*from, socket\_t \*fromlen);

返回值：ssize\_t（如果正确接收则返回接收到的字节数，失败返回-1）

函数名称：recvfrom

参数列表

int sockfd：标识一个以链接的套接字的描述字

void \*buf：接收数据缓冲区

int len：缓冲区长度

unsigned flags：调用操作方式

struct sockaddr \*from：指针，指向装有源地址的缓冲区

socket\_t \*fromlen：指针，指向from缓冲区的长度值

函数说明：用来接收远程主机经指定socket传来的数据，并把数据传到由参数buf指向的内存空间，参数len为可接收的数据的最大长度。

### 1.2 运行流程

初始化变量

|

创建套接字SOCKET

|

进入等待输入状态

|

发送输入的字符串至服务端

|

等待服务端返回信息

|

输出服务端消息到控制台

|

再次进入等待输入状态

## 2. 服务端

### 2.1 功能模块

#### 2.1.1 加载机器人聊天文本

int loadData(char \*file\_path,char \*\*input\_table,char \*\*output\_table);

返回值：int

函数名称：loadData

参数列表：

char \*file\_path：从该路径所指向的文中加载聊天数据

char \*\*input\_table：用来保存从文件中解析出来的问题列表

char \*\*out\_table：用来保存从文件中解析出来的答案列表

函数说明：从指定路径读取文件数据，通过简单的分析，生成问题列表和答案列表。

FILE\* fopen(const char\* path, const char\* mode);

返回值：FILE\*（指向指定路径的文件指针）

函数名称：fopen

参数列表

const char\* path：包含想要打开的文件路径及文件名称

const char\* mode：流形态

“r” 以[只读](http://baike.baidu.com/item/%E5%8F%AA%E8%AF%BB)方式打开文件，该文件必须存在。

“r+” 以可读写方式打开文件，该文件必须存在。

”rb+“ 读写打开一个[二进制文件](http://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%96%87%E4%BB%B6)，允许读写数据（可以任意修改），文件必须存在。

“w” 打开只写文件，若文件存在则文件长度清为0，即该文件内容会消失。若文件不存在则建立该文件。

“w+” 打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。

“a” 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。（[EOF](http://baike.baidu.com/item/EOF)符保留）

”a+“ 以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。 （原来的EOF符不保留）

“wb” 只写打开或新建一个[二进制文件](http://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%96%87%E4%BB%B6)；只允许写数据（若文件存在则文件长度清为零，即该文件内容会消失）。

“wb+” 读写打开或建立一个二进制文件，允许读和写（若文件存在则文件长度清为零，即该文件内容会消失）

“wx” 创建文本文件,只允许写入数据.[C11]

“wbx” 创建一个二进制文件,只允许写入数据.[C11]

”w+x“ 创建一个文本文件,允许读写.[C11]

“wb+x” 创建一个二进制文件,允许读写.[C11]

“w+bx” 和"wb+x"相同[C11]

“rt” 只读打开一个文本文件，只允许读数据  
“wt” 只写打开或建立一个文本文件，只允许写数据  
“at” 追加打开一个文本文件，并在文件末尾写数据  
“rb” 只读打开一个二进制文件，只允许读数据  
“wb” 只写打开或建立一个二进制文件，只允许写数据  
“ab” 追加打开一个二进制文件，并在文件末尾写数据  
“rt+” 读写打开一个文本文件，允许读和写  
“wt+” 读写打开或建立一个文本文件，允许读写  
“at+” 读写打开一个文本文件，允许读，或在文件末追加数据  
“rb+” 读写打开一个二进制文件，允许读和写  
“ab+” 读写打开一个二进制文件，允许读，或在文件末追加数据

函数说明：

char \*fgets(char \*buf, int bufsize, FILE \*stream);

返回值：char\*

1. 成功，则返回第一个参数buf；
2. 在读字符时遇到*end-of-file*，则eof指示器被设置，如果还没读入任何字符就遇到这种情况，则buf保持原来的内容，返回NULL；
3. 如果发生读入错误，error指示器被设置，返回NULL，buf的值可能被改变。

函数名称：fgets

参数列表：

char \*buf：字符型指针，指向用来存储所得数据的地址。

int bufsize：整型数据，指明存储数据的大小。

FILE \*stream：文件结构体指针，将要读取的文件流。

函数说明：

从[文件](http://baike.baidu.com/item/%E6%96%87%E4%BB%B6)结构体指针stream中读取[数据](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/33305)，每次读取一行。读取的数据保存在buf指向的字符数组中，每次最多读取bufsize-1个字符（第bufsize个字符赋'\0'），如果文件中的该行，不足bufsize个字符，则读完该行就结束。如若该行（包括最后一个换行符）的字符数超过bufsize-1，则fgets只[返回](http://baike.baidu.com/item/%E8%BF%94%E5%9B%9E/2691566)一个不完整的行，但是，缓冲区总是以NULL字符结尾，对fgets的下一次调用会继续读该行。

#### 2.1.2 产生返回消息

char \* recieve(char \*input,int size,char \*\*input\_table,char \*\*output\_table);

返回值：char\*

函数名称：recieve

参数列表：

char\* input：输入的字符数组

Int size：字符数组的长度

char \*\*input\_table：问题列表

char \*\*out\_table：答案列表

函数说明：问题列表与答案列表中的项是一一对应的关系，通过比较输入的字符数组与问题列表中的项，若能够寻找与之相对应的项，则根据下标从答案列表中取出相应的值并返回；若无法找到相对应的项，则返回字符串“I cannot understand”。

#### 2.1.3 基于UDP协议的socket的通信

int bind(int sockfd, const struct sockaddr \*my\_addr, socklen\_t addrlen);

返回值：int（成功返回0，失败则返回-1）

函数名称：bind

参数列表：

int sockfd：标识一为绑定的套接字描述字

const struct sockaddr \*my\_addr：赋予套接字的地址

socklen\_t addrlen：my\_addr的长度

函数说明：将一本地地址与一套接字绑定，bind函数通过给一个未命名套接字分配一个本地名字来为套接字建立本地捆绑。

### 2.2 运行流程

初始化变量

|

创建套接字SOCKET

|

绑定套接字（bind）

|

加载聊天文本（robot.txt）

|

等待客户端发送请求

|

接收到客户端的请求

|

分析请求，产生回复字符串

|

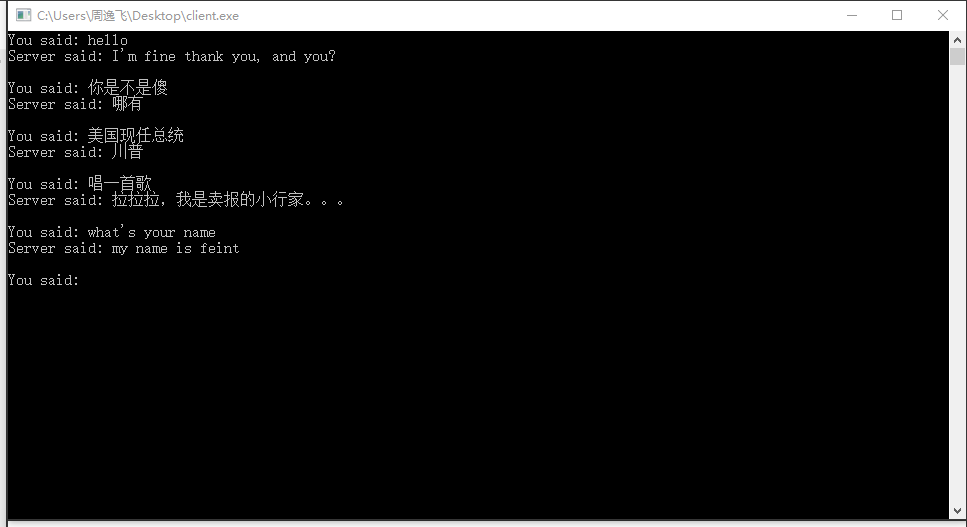
将回复信息发送给客户端

|

再次等待客户端发送请求

# 四．运行结果

客户端运行截图

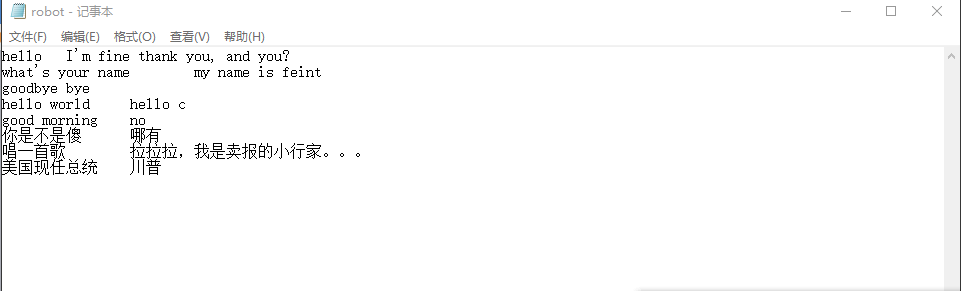


说明：首次运行客户端程序，界面显示出‘You said：’等待用户输入。输入相应的字符串后，等待服务端响应。待服务端响应之后，客户端打印出：‘Server said’＋服务端所返回的内容。

服务端运行截图



对话文本



# 五．总结

通过这次课程实践，让我接触到C语言的socket编程，在以前的日子里一直没有机会接触这个方面点知识。通过这样的一次机会，能够静下心来，花费时间去学习有关socket的知识，对C语言的使用也变的更加熟练。同时，在编写代码的过程中对面向报文的通讯协议（UDP）的理解也更加深刻，真正体会到了，在实际场景中的一个完整的运行过程。

学习新的知识，的确是一个十分有趣的过程。那种充满未知的感觉，让久久难忘。这次做的这个基于UDP协议的简易聊天机器人，除了用到socket编程的知识外，还涉及到了C语言的文件读取以及字符串的一些操作。

虽然，这次课程设计所完成的只是一个十分简单的小应用，涉及到的知识点也不是特别的多。但是，有了这次入门，也就有了继续学下去的动力，在以后的日子里，也会继续深入的学习socket编程。早日做到能够使用学习的知识编写出真正有用的程序出来。

# 六．附录

**服务端代码**

#include <stdlib.h>

#include <stdio.h>

#include <winsock2.h>

#include <time.h>

#pragma comment(lib, "ws2\_32.lib")

#define MAX\_BUFFER 1024

#define MAX\_TABLE\_ROWS 100

#define MAX\_TABLE\_COLS 256

int loadData(char \*file\_path,char \*\*input\_table,char \*\*output\_table);

char \* recieve(char \*input,int size,char \*\*input\_table,char \*\*output\_table);

int main()

{

WSADATA wsaData;

WORD sockVersion=MAKEWORD(2,2);

SOCKET server\_soc;

SOCKET client\_soc;

SOCKADDR\_IN serverAddr;

SOCKADDR\_IN clientAddr;

int server\_len=sizeof(serverAddr);

int client\_len=sizeof(clientAddr);

char \*send\_data;

int send\_len;

char recv\_buf[MAX\_BUFFER];

int result;

time\_t nowTime;

char \*\*input\_table=(char \*\*)malloc(MAX\_TABLE\_ROWS\*MAX\_TABLE\_COLS);

char \*\*output\_table=(char \*\*)malloc(MAX\_TABLE\_ROWS\*MAX\_TABLE\_COLS);

int lines=0;

serverAddr.sin\_family=AF\_INET;

serverAddr.sin\_port=htons(8088);

serverAddr.sin\_addr.s\_addr=INADDR\_ANY;

if(WSAStartup(sockVersion,&wsaData)!=0)

return 0;

//create socket

if((server\_soc=socket(AF\_INET,SOCK\_DGRAM,IPPROTO\_UDP))==INVALID\_SOCKET)

{

printf("socket error!");

return 0;

}

/\*\*\*

\* bind socket

\* bind(SOCKET,SOCKADDR \*,int SOCKADDR\_LEN)

\*\*\*/

if((result=bind(server\_soc,(SOCKADDR \*)&serverAddr,server\_len))<0)

{

printf("bind error!");

closesocket(server\_soc);

return -1;

}

printf("Server is running...\n");

client\_soc=socket(AF\_INET,SOCK\_DGRAM,0);

lines=loadData("robot.txt",input\_table,output\_table);

while(1)

{

if((result=recvfrom(server\_soc,recv\_buf,MAX\_BUFFER,

0,(SOCKADDR \*)&clientAddr,&client\_len))>=0)

{

recv\_buf[result]=0;

printf("data receive is: %s\n",recv\_buf);

send\_data=recieve(recv\_buf,lines,input\_table,output\_table);

send\_len=strlen(send\_data);

result=sendto(client\_soc,send\_data,send\_len,

0,(SOCKADDR \*)&clientAddr,client\_len);

printf("Server is running...\n");

}

}

closesocket(server\_soc);

WSACleanup();

return 0;

}

int loadData(char \*file\_path,char \*\*input\_table,char \*\*output\_table)

{

FILE \*file;

file=fopen(file\_path,"r");

int buffer\_size=256;

char \*buffer=(char\*)malloc(buffer\_size);

char strings[12][buffer\_size];

int line=0;

while(feof(file)==0)

{

fgets(strings[line++],buffer\_size,file);

}

for(int i=0; i<line; i++)

{

char \*result=strtok(strings[i],"\t");

input\_table[i]=result;

result=strtok(NULL,"\t");

output\_table[i]=result;

}

return line;

}

char \* recieve(char \*input,int size,char \*\*input\_table,char \*\*output\_table)

{

int index=-1;

for(int i=0; i<size; i++)

{

if(strcmp(input,input\_table[i])==0)

index=i;

}

if(index==-1)

return "sorry,I cannot understand!";

else

return output\_table[index];

}

**客户端代码**

#include <stdlib.h>

#include <stdio.h>

#include <winsock2.h>

#include <time.h>

#pragma comment(lib, "ws2\_32.lib")

#define MAX\_BUFFER 1024

#define SERVER\_HOST "127.0.0.1"

#define COMMON\_PORT 8088

int main()

{

WSADATA wsa;

WORD sock\_version=MAKEWORD(2,2);

SOCKADDR\_IN server\_addr;

SOCKADDR\_IN client\_addr;

SOCKET server\_soc;

SOCKET client\_soc;

int client\_len=sizeof(client\_addr);

int server\_len=sizeof(server\_addr);

char recv\_buf[MAX\_BUFFER];

int time\_out=2000;

int result;

char \*send\_data=(char \*)malloc(MAX\_BUFFER);

if(WSAStartup(sock\_version,&wsa)!=0)

return 0;

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(COMMON\_PORT);

server\_addr.sin\_addr.s\_addr=inet\_addr(SERVER\_HOST);

if((server\_soc=socket(AF\_INET,SOCK\_DGRAM,0))==INVALID\_SOCKET)

{

printf("socket error!");

return -1;

}

result=setsockopt(server\_soc,SOL\_SOCKET,SO\_RCVTIMEO,(char \*)&time\_out,sizeof(time\_out));

client\_soc=socket(AF\_INET,SOCK\_DGRAM,0);

while(1)

{

// scanf("%s",send\_data);

memset(send\_data,0,sizeof(send\_data));

memset(recv\_buf,0,MAX\_BUFFER);

printf("You said: ");

gets(send\_data);

result=sendto(server\_soc,send\_data,strlen(send\_data),

0,(SOCKADDR \*)&server\_addr,server\_len);

result=recvfrom(server\_soc,recv\_buf,MAX\_BUFFER,0,(SOCKADDR \*)&client\_addr,&client\_len);

// printf("result: %d\n",result);

if(result >=0)

{

recv\_buf[result]=0;

printf("Server said: %s\n",recv\_buf);

}

}

closesocket(server\_soc);

WSACleanup();

return 0;

}