[请输入论文题目]

**[请输入学校名称]毕业论文模板**

专业：[请 输 入 专 业]

学生姓名：[请 输 入 姓 名]

指导教师：[请输入指导教师]

完成时间：2011年3月2日

**需求分析**

**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。

**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**

**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，对界面的友好程度要求不高

**2.1 系统范围**

**2.2 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。

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。

**2.3 需求分析**

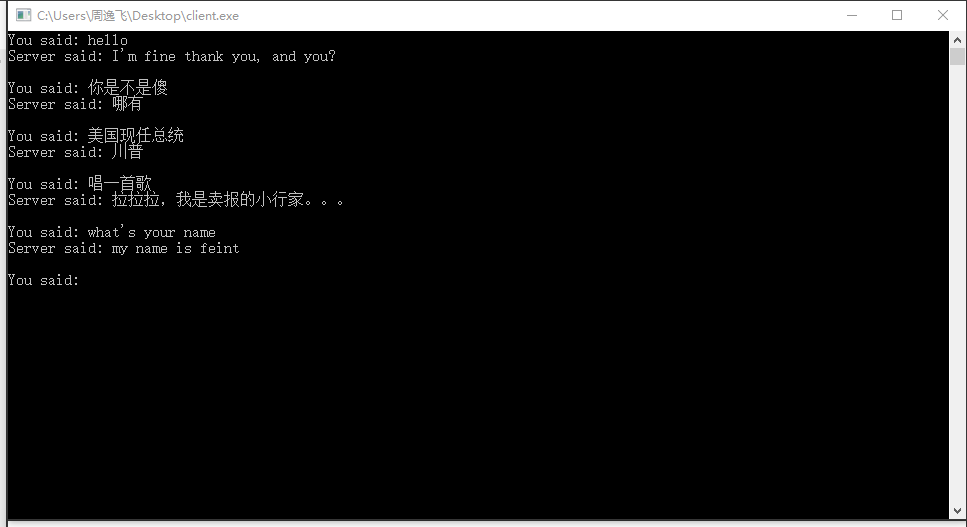
**2.3.1 与机器人进行交流**

**2.3.1.1 功能描述**

在客户端程序中，输入文字之后，系统会进行相应的回答。以此来模拟与机器人进行简易的对话。

**2.3.1.2 用例描述**

**2.3.1.3 用户界面**

****

**概要设计**

**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**

**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

1. **服务端**

**2.1 功能模块**

**2.1.1 加载机器人聊天文本**

**2.1.2 产生返回消息**

**2.1.3 基于UDP协议的socket的通信**

**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

|

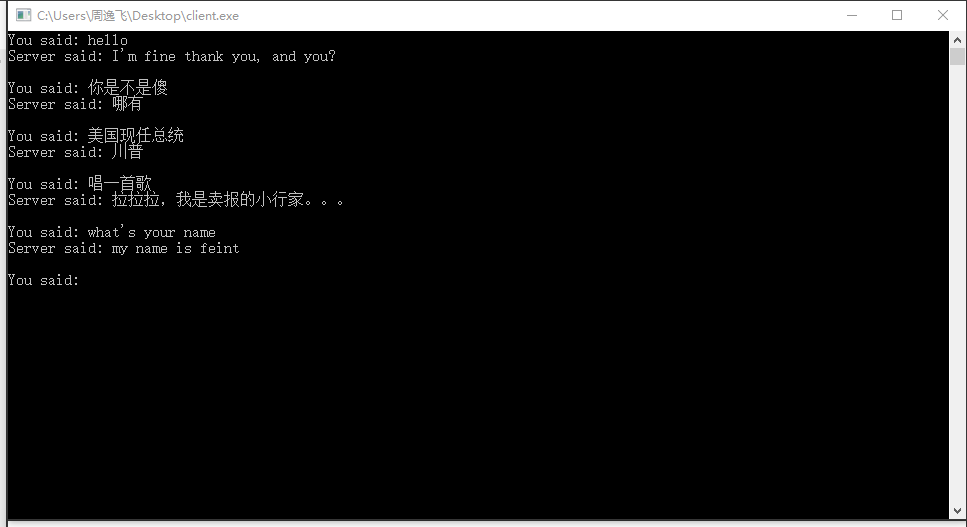
将回复信息发送给客户端

|

再次等待客户端发送请求

**运行结果**

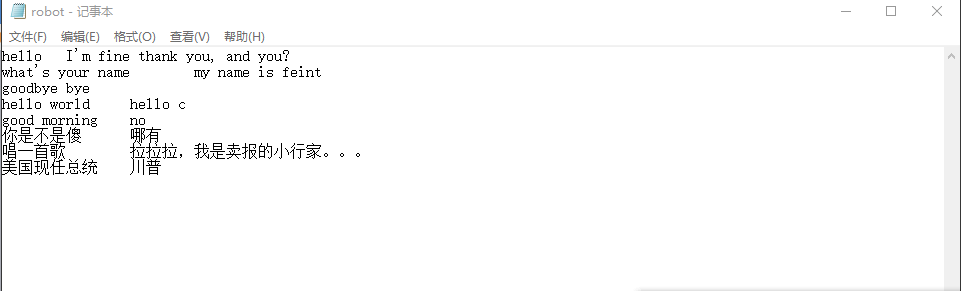
客户端运行截图

****

服务端运行截图

****

对话文本

****

**总结**

通过这次课程实践，让我接触到C语言的socket编程，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。

**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;

}