1. UDP客户端套接字的地址分配

UDP程序中， 调用sendto函数传输数据前完成套接字地址分配工作。

可以调用bind函数，bind函数不区分是TCP还是UDP。

如果调用sendto函数时发现尚未分配地址，则首次调用sendto时自动分配IP和端口，并且分配的地址移植保留到程序结束。

2. UDP存在数据边界

与TCP不一样，UDP协议中，调用IO函数的次数很重要，输入函数的调用次数和输出函数的调用次数完全一致，才能保证接收完全。

其实，UDP套接字传输的数据包又称为数据报，它本身可以成为一个完整数据。（一次性发出）

3.举例bound\_host：host2程序调用3次sendto传输数据，host1调用3次recvfrom函数接收数据。

Host1 代码（这里充当服务器）：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#define BUF\_SIZE 30

void error\_handling(char \*message);

int main(int argc, char \*argv[]){

int sock;

char message[BUF\_SIZE];

struct sockaddr\_in my\_adr, your\_adr;

socklen\_t adr\_sz;

int str\_len,i;

if(argc != 2){

printf("Usage : %s <port> \n",argv[0]);

exit(1);

}

sock = socket(PF\_INET,SOCK\_DGRAM,0);

if(sock == -1){

error\_handling("socket error");

}

memset(&my\_adr,0,sizeof(my\_adr));

my\_adr.sin\_family = AF\_INET;

my\_adr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

my\_adr.sin\_port = htons(atoi(argv[1]));

if(bind(sock,(struct sockaddr \*)&my\_adr,sizeof(my\_adr)) == -1){

error\_handling("bind error");

}

for(i=0;i<3;i++){

sleep(5);

adr\_sz = sizeof(your\_adr);

str\_len = recvfrom(sock, message, BUF\_SIZE,0,(struct sockaddr\*)&your\_adr,&adr\_sz);

//your\_adr这个地址需要从客户端传递过来，然后在这里传递给该变量

printf("Message %d: %s \n",i+1,message);

}

close(sock);

return 0;

}

void error\_handling(char \*message){

fputs(message,stderr);

fputs("\n",stderr);

exit(1);

}

Host2端代码（这里充当客户端）：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#define BUF\_SIZE 30

void error\_handling(char \*message);

int main(int argc, char \*argv[]){

int sock;

char msg1[] = "Hi!";

char msg2[] = "I'm another UDP host!";

char msg3[] = "Nice to meet you";

struct sockaddr\_in your\_adr;

socklen\_t your\_adr\_sz;

if(argc != 3){

printf("Usage : %s <IP><port> \n",argv[0]);

exit(1);

}

sock = socket(PF\_INET,SOCK\_DGRAM,0);

if(sock == -1){

error\_handling("socket error");

}

//your\_adr 通过客户端的命令行传递过来

memset(&your\_adr,0,sizeof(your\_adr));

your\_adr.sin\_family = AF\_INET;

your\_adr.sin\_addr.s\_addr = inet\_addr(argv[1]);

your\_adr.sin\_port = htons(atoi(argv[2]));

sendto(sock,msg1,sizeof(msg1),0,(struct sockaddr\*)&your\_adr,sizeof(your\_adr));

sendto(sock,msg2,sizeof(msg2),0,(struct sockaddr\*)&your\_adr,sizeof(your\_adr));

sendto(sock,msg3,sizeof(msg3),0,(struct sockaddr\*)&your\_adr,sizeof(your\_adr));

close(sock);

return 0;

}

void error\_handling(char \*message){

fputs(message,stderr);

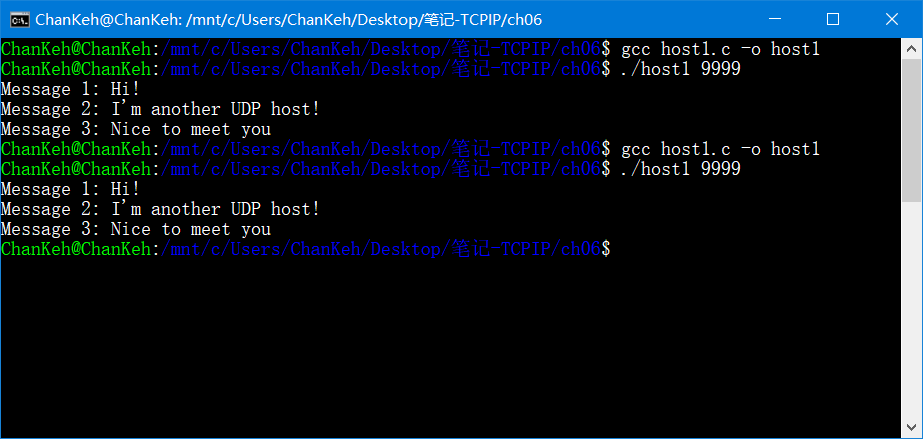
fputs("\n",stderr);

exit(1);

}

运行结果：

服务器端：



客户端：

