UDP协议则是一种无连接的，不可靠的数据报（SOCK\_DGRAM）传输服务。使用UDP套接口不用建立连接，服务端在调用socket()生成一个套接字并调用bind()绑定端口后就可以进行通信（recvfrom函数和sendto函数）了；客户端在用socket()生成一个套接字后就可以向服务端地址发送和接收数据了。

此处需要特别注意：TCP使用的是流套接字（SOCK\_STREAM），UDP使用的是数据报套接字（SOCK\_DGRAM）

UDP套接字编程范例：

server端代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> File Name: server.c

> Author: SongLee

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<sys/types.h>

#include<sys/socket.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<stdlib.h>

#include<errno.h>

#include<netdb.h>

#include<stdarg.h>

#include<string.h>

#define SERVER\_PORT 8000

#define BUFFER\_SIZE 1024

#define FILE\_NAME\_MAX\_SIZE 512

int main()

{

/\* 创建UDP套接口 \*/

struct sockaddr\_in server\_addr;

bzero(&server\_addr, sizeof(server\_addr));

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

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

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

/\* 创建socket \*/

int server\_socket\_fd = socket(AF\_INET, SOCK\_DGRAM, 0);

if(server\_socket\_fd == -1)

{

perror("Create Socket Failed:");

exit(1);

}

/\* 绑定套接口 \*/

if(-1 == (bind(server\_socket\_fd,(struct sockaddr\*)&server\_addr,sizeof(server\_addr))))

{

perror("Server Bind Failed:");

exit(1);

}

/\* 数据传输 \*/

while(1)

{

/\* 定义一个地址，用于捕获客户端地址 \*/

struct sockaddr\_in client\_addr;

socklen\_t client\_addr\_length = sizeof(client\_addr);

/\* 接收数据 \*/

char buffer[BUFFER\_SIZE];

bzero(buffer, BUFFER\_SIZE);

if(recvfrom(server\_socket\_fd, buffer, BUFFER\_SIZE,0,(struct sockaddr\*)&client\_addr, &client\_addr\_length) == -1)

{

perror("Receive Data Failed:");

exit(1);

}

/\* 从buffer中拷贝出file\_name \*/

char file\_name[FILE\_NAME\_MAX\_SIZE+1];

bzero(file\_name,FILE\_NAME\_MAX\_SIZE+1);

strncpy(file\_name, buffer, strlen(buffer)>FILE\_NAME\_MAX\_SIZE?FILE\_NAME\_MAX\_SIZE:strlen(buffer));

printf("%s\n", file\_name);

}

close(server\_socket\_fd);

return 0;

}

client端代码如下：

?

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> File Name: client.c

> Author: SongLee

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<sys/types.h>

#include<sys/socket.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<stdlib.h>

#include<errno.h>

#include<netdb.h>

#include<stdarg.h>

#include<string.h>

#define SERVER\_PORT 8000

#define BUFFER\_SIZE 1024

#define FILE\_NAME\_MAX\_SIZE 512

int main()

{

/\* 服务端地址 \*/

struct sockaddr\_in server\_addr;

bzero(&server\_addr, sizeof(server\_addr));

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

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

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

/\* 创建socket \*/

int client\_socket\_fd = socket(AF\_INET, SOCK\_DGRAM, 0);

if(client\_socket\_fd < 0)

{

perror("Create Socket Failed:");

exit(1);

}

/\* 输入文件名到缓冲区 \*/

char file\_name[FILE\_NAME\_MAX\_SIZE+1];

bzero(file\_name, FILE\_NAME\_MAX\_SIZE+1);

printf("Please Input File Name On Server:\t");

scanf("%s", file\_name);

char buffer[BUFFER\_SIZE];

bzero(buffer, BUFFER\_SIZE);

strncpy(buffer, file\_name, strlen(file\_name)>BUFFER\_SIZE?BUFFER\_SIZE:strlen(file\_name));

/\* 发送文件名 \*/

if(sendto(client\_socket\_fd, buffer, BUFFER\_SIZE,0,(struct sockaddr\*)&server\_addr,sizeof(server\_addr)) < 0)

{

perror("Send File Name Failed:");

exit(1);

}

close(client\_socket\_fd);

return 0;

}