通信过程中一般都需要提前定义好通信协议，TCP中亦是如此。

我们编写计算器的例子，定义好应用层协议：

    1.客户端连接到服务器端后以1字节整数形式传递待算数字个数；

    2.客户端向服务器端传递的每个整数型数据占用4字节；

    3.传递完数据后，接着1个字节运算符；

    4.选择字符+，-，\*，/;

    5.服务器以4字节整数型向客户端传回运算结果；

    6.客户端得到运算结果后终止与服务器端的连接。



Server端：

1. #include **<stdio.h>**
2. #include **<stdlib.h>**
3. #include **<string.h>**
4. #include **<unistd.h>**
5. #include **<arpa**/inet.h**>**
6. #include **<sys**/socket.h**>**
8. #define BUF\_SIZE    1024
9. #define OPSZ    4
11. void error\_handling(char \*message);
12. int calculate(int opnum,int opnds[],char oprator);
14. int main(int argc, char \*argv[]){
15. int serv\_sock;
16. int clnt\_sock;
17. char opinfo[BUF\_SIZE];
18. int result,opnd\_cnt,i;
19. int recv\_cnt,recv\_len;
20. struct sockaddr\_in serv\_addr;
21. struct sockaddr\_in clnt\_addr;
22. socklen\_t clnt\_addr\_sz;
24. if(argc != 2){
25. printf("Usage : %s **<port>**\n",argv[0]);
26. exit(1);
27. }
28. serv\_sock = socket(PF\_INET,SOCK\_STREAM,0);
29. if(serv\_sock == -1){
30. error\_handling("socket() error");
31. }
33. memset(&serv\_addr,0,sizeof(serv\_addr));
34. serv\_addr.sin\_family=AF\_INET;
35. serv\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);
36. serv\_addr.sin\_port=htons(atoi(argv[1]));
38. if(bind(serv\_sock,(struct sockaddr\*)&serv\_addr,sizeof(serv\_addr)) == -1){
39. error\_handling("bind() error");
40. }
42. if(listen(serv\_sock,5) == -1){
43. error\_handling("listen() error");
44. }
46. clnt\_addr\_sz = sizeof(clnt\_addr);
48. for(i=0;i**<5**;i++){
49. opnd\_cnt = 0;
50. clnt\_sock = accept(serv\_sock,(struct sockaddr \*)&clnt\_addr,&clnt\_addr\_sz);
51. read(clnt\_sock,&opnd\_cnt,1);  //读取操作数个数，用一个字节存储
53. recv\_len = 0;
54. while((opnd\_cnt\*OPSZ+1)**>**recv\_len){ //跳过运算数个数那个字节
55. recv\_cnt = read(clnt\_sock,&opinfo[recv\_len],BUF\_SIZE-1);
56. //接收的数据存放到opinfo缓冲中去
57. recv\_len += recv\_cnt;
58. }
59. //每次读取 recv\_cnt 个字节的数据，一直到opnd\_cnt\*OPSZ+1，表示从客户端接收完毕
60. result = calculate(opnd\_cnt,(int \*)opinfo,opinfo[recv\_len-1]);
61. write(clnt\_sock,(char \*)&result,sizeof(result));
62. //将结果通过客户端请求套接字的发送缓冲区发送出去，传给客户端
63. close(clnt\_sock);
64. }
65. close(serv\_sock);
67. return 0;
69. }
71. int calculate(int opnum,int opnds[],char op){
72. int result = opnds[0],i;  //因为读取的时候，已经跳过了操作数个数那个字节
73. //所以，这里直接读取下标为0的数就为第一个操作数
74. switch(op){
75. case '+':
76. for(i=1;i**<opnum**;i++) result +=opnds[i];
77. break;
78. case '-':
79. for(i=1;i**<opnum**;i++) result -= opnds[i];
80. break;
81. case '\*':
82. for(i=1;i**<opnum**;i++) result \*= opnds[i];
83. break;
84. }
85. return result;
86. }

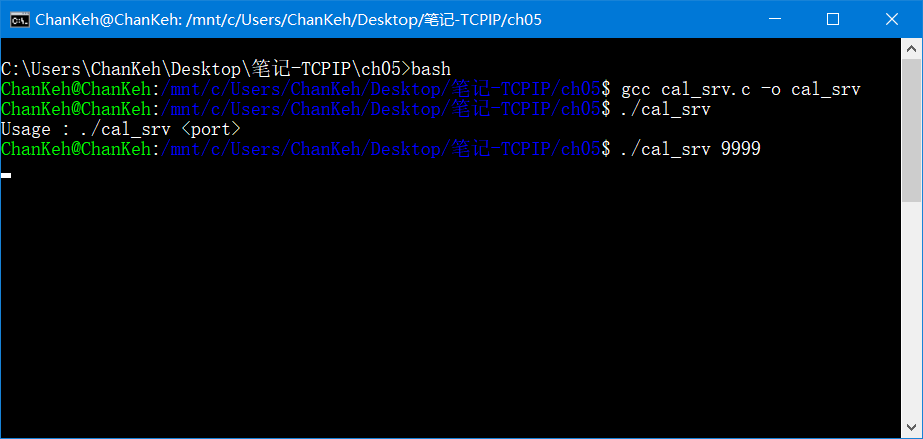

90. void error\_handling(char \*message){
92. fputs(message,stderr);
93. fputs("\n",stderr);
94. exit(1);
95. }

Client端：

1. #include **<stdio.h>**
2. #include **<stdlib.h>**
3. #include **<string.h>**
4. #include **<unistd.h>**
5. #include **<arpa**/inet.h**>**
6. #include **<sys**/socket.h**>**
8. #define BUF\_SIZE    1024
9. #define RLT\_SIZE    4
10. #define OPSZ        4
12. void error\_handling(char \*message);
14. int main(int argc,char \*argv[]){
15. int sock;
16. char opmsg[BUF\_SIZE];
17. int result,opnd\_cnt,i;
18. struct sockaddr\_in serv\_addr;
20. if(argc != 3){
21. printf("Usage : %s **<IP>** **<port>**\n",argv[0]);
22. exit(1);
23. }
25. sock = socket(PF\_INET,SOCK\_STREAM,0);
26. if(sock == -1){
27. error\_handling("socket() error");
28. }
30. memset(&serv\_addr,0,sizeof(serv\_addr));
31. serv\_addr.sin\_family = AF\_INET;
32. serv\_addr.sin\_addr.s\_addr = inet\_addr(argv[1]);
33. serv\_addr.sin\_port = htons(atoi(argv[2]));
35. if(connect(sock,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr)) == -1){
36. error\_handling("connect() error\r\n");
37. }else{
38. printf("Connected....");
39. }
41. fputs("Operand count: ",stdout);
42. scanf("%d",&opnd\_cnt);
43. opmsg[0] = (char)opnd\_cnt;//文章一开始所说的，用一个字节存储操作数个数
44. for(i=0;i**<opnd\_cnt**;i++){
45. printf("Operand %d: ",i+1);
46. scanf("%d",(int \*)&opmsg[i\*OPSZ+1]);//+1的目的是为了跳过第一个字节
47. }
48. fgetc(stdin);   //end char ，去掉\n
49. fputs("Operator: ",stdout);
50. scanf("%c",&opmsg[opnd\_cnt\*OPSZ+1]);//输入运算符
51. write(sock,opmsg,opnd\_cnt\*OPSZ+2); //一次性传输数组中的数据
52. read(sock,&result,RLT\_SIZE);  //读取结果
54. printf("Operation result:%d \n",result);
55. close(sock);
56. return 0;
57. }
59. void error\_handling(char \*message){
60. fputs(message,stderr);
61. fputs("\n",stderr);
62. exit(1);
63. }

显示结果：

服务器端：



客户端：

