1. 套接字数据通信中使用标准IO函数：

a. 具有良好的移植性portability；

//所有标准函数都具有良好的移植性

b. 标准IO可以利用缓冲提高性能；

2. 使用标准IO的几个缺陷

a. 不容易进行双向通信；

//同时读写方式打开r+,w+,a+

b. 有时候可能频繁调用fflush；

//每次切换读写都应该调用fflush

c. 需要以FILE结构体指针形式返回文件描述符；

//需要文件描述符转成FILE的方法

3. fdopen将文件描述符转换成FILE结构体指针

1. #include <stdio.h>
3. **FILE** \* fdopen(**int** fildes, **const** **char** \*mode);
5. 成功返回转换的**FILE**结构体指针，失败NULL
7. fildes ：需要转换的文件描述符
8. mode：将要创建的**FILE**结构体指针的mode信息 //与fopen的打开模式相同

4. fileno将FILE结构体指针转换成文件描述符

1. #include <stdio.h>
3. **int** fileno(**FILE** \* stream);
5. 成功返回转换后的文件描述符，失败-1

5. 使用标准IO实现echo服务器/客户端

server：

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
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** serv\_sock,clnt\_sock;
14. **char** message[BUF\_SIZE];
15. **int** str\_len,i;
16. **struct** sockaddr\_in serv\_adr;
17. **struct** sockaddr\_in clnt\_adr;
18. socklen\_t clnt\_adr\_sz;
19. **FILE** \* readfp;
20. **FILE** \* writefp;
22. **if**(argc != 2){
23. printf("Usage : %s <port>\n",argv[0]);
24. exit(1);
25. }
27. serv\_sock = socket(PF\_INET,SOCK\_STREAM,0);
28. **if**(serv\_sock == -1){
29. error\_handling("socket() error");
30. }
32. memset(&serv\_adr,0,**sizeof**(serv\_adr));
33. serv\_adr.sin\_family=AF\_INET;
34. serv\_adr.sin\_addr.s\_addr=htonl(INADDR\_ANY);
35. serv\_adr.sin\_port=htons(atoi(argv[1]));
37. **if**(bind(serv\_sock,(**struct** sockaddr\*)&serv\_adr,**sizeof**(serv\_adr)) == -1){
38. error\_handling("bind() error");
39. }
41. **if**(listen(serv\_sock,5) == -1){
42. error\_handling("listen error");
43. }
45. clnt\_adr\_sz = **sizeof**(clnt\_adr);
46. **for**(i=0;i<5;i++){
47. clnt\_sock = accept(serv\_sock,(**struct** sockaddr\*)&clnt\_adr,&clnt\_adr\_sz);
48. **if**(clnt\_sock == -1){
49. error\_handling("accept error");
50. }**else**{
51. printf("connected client %d %d\n",i+1,clnt\_sock);
52. }
54. readfp = fdopen(clnt\_sock,"r");  //文件描述符转换成文件指针
55. writefp = fdopen(clnt\_sock,"w");  //文件描述符转换成文件指针
57. **while**(!feof(readfp)){
58. fgets(message,BUF\_SIZE,readfp);
59. fputs(message,writefp);
60. fflush(writefp);  //对于文件指针而言，每次都要fflush，刷新缓冲
61. }
63. printf("disconnected client %d \n",clnt\_sock);
64. fclose(readfp);
65. fclose(writefp);
66. }
68. close(serv\_sock);
70. **return** 0;
72. }
74. **void** error\_handling(**char** \*message){
76. fputs(message,stderr);
77. fputs("\n",stderr);
78. exit(1);
79. }

client：

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
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** sock;
14. **char** message[BUF\_SIZE];
15. **int** str\_len;
16. **struct** sockaddr\_in serv\_adr;
17. **FILE** \* readfp;
18. **FILE** \* writefp;
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\_adr,0,**sizeof**(serv\_adr));
31. serv\_adr.sin\_family=AF\_INET;
32. serv\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]);
33. serv\_adr.sin\_port=htons(atoi(argv[2]));
35. **if**(connect(sock,(**struct** sockaddr \*)&serv\_adr,**sizeof**(serv\_adr)) == -1){
36. error\_handling("connect error");
37. }**else** {
38. puts("connected...");
39. }
41. readfp = fdopen(sock,"r"); //文件描述符转成文件指针
42. writefp = fdopen(sock,"w");  //+1
44. **while**(1){
45. fputs("Insert message(q to quit):",stdout);
46. fgets(message,BUF\_SIZE,stdin);//从控制台输入到messsage中
47. **if**(!strcmp(message,"q\n") || !strcmp(message,"Q\n"))
48. **break**;
49. fputs(message,writefp);  //messag缓冲数据存放到文件中
50. fflush(writefp);//刷新文件缓冲
51. fgets(message,BUF\_SIZE,readfp);
52. printf("Message from server: %s",message);
53. }
55. fclose(writefp);
56. fclose(readfp);
58. **return** 0;
59. }
61. **void** error\_handling(**char** \*message){
63. fputs(message,stderr);
64. fputs("\n",stderr);
65. exit(1);
66. }