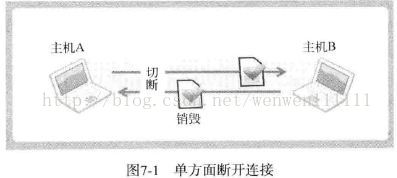
由于之前曾经说过，sock套接字即使断开连接了，只要发送缓冲里仍有数据，依然会发出，尽管发出的数据无法被对方接收了，使用close后会关闭套接字，但仍有很多问题，比如发送缓冲中仍然有数据，对方还没有完全接收。

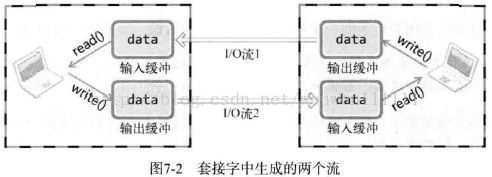
1.使用close，closesocket（windows下）函数是单方面断开连接，不够优雅。调用了close意味着完全断开连接，不能发送也不能接收数据。



图中，A发送完数据后断开连接，之后主机再也不发接收B的数据，如此不够优雅

2. 使用shutdown可以关闭其中的一个流。

1. #include <sys/socket.h>
3. **int** shutdown(**int** sock, **int** howto);
4. 成功返回0，失败返回-1
6. sock：需要断开的套接字文件描述符
7. howto：断开方式
8. SHUT\_RD：断开输入流
9. SHUT\_WR：断开输出流
10. SHUT\_RDWR：同时断开IO流



3. 基于半关闭的文件传输



file\_client.c程序：

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    30
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** sd;
14. **FILE** \*fp;
15. **char** buf[BUF\_SIZE];
16. **int** read\_cnt;
17. **struct** sockaddr\_in serv\_adr;
19. **if**(argc != 3){
20. printf("Usage : %s <IP> <port>\n",argv[0]);
21. exit(1);
22. }
24. fp = fopen("receive.dat","wb");
26. sd = socket(PF\_INET,SOCK\_STREAM,0);
27. **if**(sd == -1){
28. error\_handling("socket() error");
29. }
31. memset(&serv\_adr,0,**sizeof**(serv\_adr));
32. serv\_adr.sin\_family=AF\_INET;
33. serv\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]);
34. serv\_adr.sin\_port=htons(atoi(argv[2]));
36. connect(sd,(**struct** sockaddr \*)&serv\_adr,**sizeof**(serv\_adr));
38. **while**((read\_cnt = read(sd,buf,BUF\_SIZE)) !=0 ){
39. fwrite((**void** \*)buf,1,read\_cnt,fp);
40. }
41. shutdown(sd,SHUT\_RD);//关闭读，仅能写
42. puts("Received file data");
43. write(sd,"Thank you",10);
44. fclose(fp);
45. close(sd);
47. **return** 0;
49. }
51. **void** error\_handling(**char** \*message){
53. fputs(message,stderr);
54. fputs("\n",stderr);
55. exit(1);
56. }

file\_server.c程序：

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    30
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** serv\_sd,clnt\_sd;
14. **FILE** \*fp;
15. **char** buf[BUF\_SIZE];
16. **int** read\_cnt;
17. **struct** sockaddr\_in serv\_adr;
18. **struct** sockaddr\_in clnt\_adr;
19. socklen\_t clnt\_adr\_sz;
21. **if**(argc != 2){
22. printf("Usage : %s <port>\n",argv[0]);
23. exit(1);
24. }
26. fp = fopen("file\_server.c","rb");
27. serv\_sd = socket(PF\_INET,SOCK\_STREAM,0);
28. **if**(serv\_sd == -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\_sd,(**struct** sockaddr\*)&serv\_adr,**sizeof**(serv\_adr)) == -1){
38. error\_handling("bind() error");
39. }
41. listen(serv\_sd,5);
43. clnt\_adr\_sz = **sizeof**(clnt\_adr);
44. clnt\_sd = accept(serv\_sd,(**struct** sockaddr \*)&clnt\_adr,&clnt\_adr\_sz);
46. **while**(1){
47. read\_cnt = fread((**void** \*)buf,1,BUF\_SIZE,fp);
48. **if**(read\_cnt<BUF\_SIZE){
49. write(clnt\_sd,buf,read\_cnt);
50. **break**;
51. }
52. write(clnt\_sd,buf,BUF\_SIZE);
53. }
55. shutdown(clnt\_sd,SHUT\_WR);  //进入半关闭状态，关闭外写流，仅能读取
56. read(clnt\_sd,buf,BUF\_SIZE);
57. printf("Message from client:%s \n",buf);
59. fclose(fp);
60. close(clnt\_sd);
61. close(serv\_sd);
62. **return** 0;
63. }
65. **void** error\_handling(**char** \*message){
67. fputs(message,stderr);
68. fputs("\n",stderr);
69. exit(1);
70. }