1. IO流分离方式（两种）：

a. 第十章TCP IO routine分离，通过fork文件描述符区分输入输出，虽然文件描述符不会根据输入、输出进行区分，但是分开了两个文件描述符的用途（父进程负责读，子进程负责写）；

b. 第十五章 调用fdopen创建FILE指针，分离输入工具和输出工具。

2. 分离流的好处：

//第十章

a. 通过分开输入过程和输出过程降低实现难度；

b. 与输入无关的输出操作可以提高速度；

//第十五章

a. 将FILE指针按读模式、写模式加以区分；

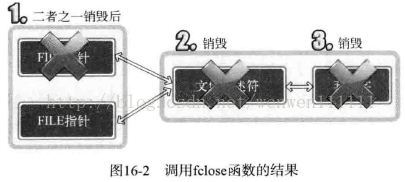
b. 可以通过区分读写模式降低难度；

c. 通过区分IO缓冲提高缓冲性能；

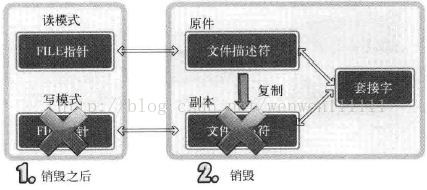
d. 将文件描述符转成文件指针后，可以使用标准IO函数

3. 第十五章流分离的问题

读模式FILE指针、写模式FILE指针都是基于同一个文件描述符创建的，fclose关闭任意一个，都将关闭文件描述符，套接字终止。



4. 解决如上问题，只需在创建FILE指针前复制文件描述符。



复制文件描述符的方法dup/dup2:

1. #include <unistd.h>
3. **int** dup(**int** fildes)
4. **int** dup2(**int** fildes, **int** fildes2)
6. 成功时返回复制的文件描述符，失败返回-1
8. fildes：需要复制的文件描述符
9. filders2：明确指定的文件描述符整数值

5. 复制文件描述符进行半关闭示例

server.c：

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>
7. #define BUF\_SIZE 1024
9. **int** main(**int** argc, **char** \* argv[]){
10. **int** serv\_sock,clnt\_sock;
11. **FILE** \*readfp;
12. **FILE** \*writefp;
13. **struct** sockaddr\_in serv\_adr,clnt\_adr;
14. socklen\_t clnt\_adr\_sz;
15. **char** buf[BUF\_SIZE];
17. serv\_sock = socket(PF\_INET,SOCK\_STREAM,0);
19. memset(&serv\_adr,0,**sizeof**(serv\_adr));
20. serv\_adr.sin\_family = AF\_INET;
21. serv\_adr.sin\_addr.s\_addr = htonl(INADDR\_ANY);
22. serv\_adr.sin\_port = htons(atoi(argv[1]));
24. bind(serv\_sock,(**struct** sockaddr \*)&serv\_adr,**sizeof**(serv\_adr));
26. listen(serv\_sock,5);
28. clnt\_adr\_sz = **sizeof**(clnt\_adr);
30. clnt\_sock = accept(serv\_sock,(**struct** sockaddr \*)&clnt\_adr,&clnt\_adr\_sz);
32. readfp = fdopen(clnt\_sock,"r");
33. writefp = fdopen(dup(clnt\_sock),"w");
35. fputs("from server: Hi~ client? \n",writefp);
36. fputs("I love all the world! \n",writefp);
37. fflush(writefp);
39. shutdown(fileno(writefp),SHUT\_WR);
41. fclose(writefp);
42. fgets(buf,**sizeof**(buf),readfp);
43. fputs(buf,stdout);
45. fclose(readfp);
47. **return** 0;
48. }

client.c：

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. **int** main(**int** argc, **char** \* argv[]){
11. **int** sock;
12. **FILE** \*readfp;
13. **FILE** \*writefp;
14. **struct** sockaddr\_in serv\_adr;
15. socklen\_t clnt\_adr\_sz;
16. **char** buf[BUF\_SIZE];
18. sock = socket(PF\_INET,SOCK\_STREAM,0);
20. memset(&serv\_adr,0,**sizeof**(serv\_adr));
21. serv\_adr.sin\_family = AF\_INET;
22. serv\_adr.sin\_addr.s\_addr = inet\_addr(argv[1]);
23. serv\_adr.sin\_port = htons(atoi(argv[2]));
25. connect(sock,(**struct** sockaddr \*)&serv\_adr,**sizeof**(serv\_adr));
27. readfp = fdopen(sock,"r");
28. writefp = fdopen(sock,"w");
30. **while**(1){
31. **if**(fgets(buf,**sizeof**(buf),readfp) == NULL){
32. **break**;
33. }
35. fputs(buf,stdout);
36. fflush(stdout);
37. }
39. fputs("from client: thank you\n",writefp);
40. fflush(writefp);
41. fclose(readfp);
42. fclose(writefp);
44. **return** 0;
45. }