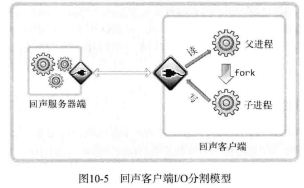
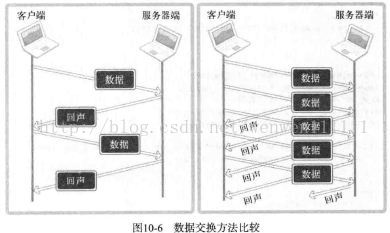
1. 分割IO程序：将接收数据，发送数据的进程分开，使程序的实现变得更加简单



2. 分割IO的实现模型



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);
11. **void** read\_routine(**int** sock,**char** \*buf);
12. **void** write\_routine(**int** sock,**char** \*buf);
14. **int** main(**int** argc,**char** \*argv[]){
15. **int** sock;
16. pid\_t pid;
17. **struct** sockaddr\_in serv\_adr;
18. **char** buf[BUF\_SIZE];
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\r\n");
37. }**else**{
38. printf("Connected....");
39. }
41. pid = fork();
43. **if**(pid == 0){
44. write\_routine(sock,buf);  //子进程负责数据的写入
45. }**else**{
46. read\_routine(sock,buf);  //父进程负责数据的读
47. }
48. close(sock);
49. **return** 0;
50. }
52. **void** error\_handling(**char** \*message){
53. fputs(message,stderr);
54. fputs("\n",stderr);
55. exit(1);
56. }
58. **void** read\_routine(**int** sock, **char** \*buf){
59. **while**(1){
60. **int** str\_len = read(sock,buf,BUF\_SIZE);
61. **if**(str\_len == 0){
62. **return** ;
63. }
65. buf[str\_len] = 0;
66. printf("Message from server : %s ",buf);
67. }
68. }
70. **void** write\_routine(**int** sock, **char** \*buf){
71. **while**(1){
72. fgets(buf,BUF\_SIZE,stdin);
73. **if**(!strcmp(buf,"q\n") || !strcmp(buf,"Q\n")){
74. shutdown(sock,SHUT\_WR);
75. **return** ;
76. }
77. write(sock,buf,strlen(buf));
78. }
79. }