1. 在UDP中，使用sendto数据传输过程分三阶段：

a. 向UDP套接字注册目标IP和端口号；

b. 传输数据；

c. 删除UDP套接字中注册的目标地址信息。

2. unconnected UDP套接字：

每次调用sendto重复1的三个步骤，每次都变更目标地址，因此可以重复利用同一UDP向不同目标传递数据。

注册了目标地址的UDP套接字是connected UDP，默认情况下是unconnected的。需要与同一主机进行长时间通信时，将套接字变成connected会提高效率。

Connected UDP套接字不仅可以使用sendto和recvfrom函数，甚至还可以使用write和read函数。

3. connected套接字：

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** sock;
14. **char** message[BUF\_SIZE];
15. **int** str\_len;
16. socklen\_t adr\_sz;
17. **struct** sockaddr\_in serv\_adr, from\_adr;
19. **if**(argc != 3){
20. printf("Usage : %s <IP> <port>\n",argv[0]);
21. exit(1);
22. }
24. sock = socket(PF\_INET,SOCK\_DGRAM,0);
25. **if**(sock == -1){
26. error\_handling("socket() error");
27. }
29. memset(&serv\_adr,0,**sizeof**(serv\_adr));
30. serv\_adr.sin\_family=AF\_INET;
31. serv\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]);
32. serv\_adr.sin\_port=htons(atoi(argv[2]));
34. connect(sock,(**struct** sockaddr\*)&serv\_adr,**sizeof**(serv\_adr));
35. //后面可以直接使用read和write函数进行操作
36. **while**(1){
38. fputs("Insert message(q to quit):",stdout);
39. fgets(message,**sizeof**(message),stdin);
40. **if**(!strcmp(message,"q\n") || !strcmp(message,"Q\n"))
41. **break**;
42. //        sendto(sock,message,strlen(message),0,(struct sockaddr \*)&serv\_adr,sizeof(serv\_adr));
43. write(sock,message,strlen(message));
44. //adr\_sz = sizeof(from\_adr);
45. //str\_len = recvfrom(sock,message,BUF\_SIZE,0,(struct sockaddr \*)&from\_adr,&adr\_sz);
46. str\_len = read(sock,message,**sizeof**(message)-1);
47. message[str\_len] = 0;
49. printf("Message from server: %s",message);
50. }
52. close(sock);
54. **return** 0;
56. }
58. **void** error\_handling(**char** \*message){
60. fputs(message,stderr);
61. fputs("\n",stderr);
62. exit(1);
63. }