code/netp/echoserveri.c

```
1  #include "csapp.h"
2
3  void echo(int connfd);
4
5  int main(int argc, char **argv)
6  {
7      int listenfd, connfd;
8      socklen_t clientlen;
9      struct sockaddr_storage clientaddr;  /* Enough space for any address */
10     char client_hostname[MAXLINE], client_port[MAXLINE];
11
12     if (argc != 2) {
13         fprintf(stderr, "usage: %s <port>\n", argv[0]);
14         exit(0);
15     }
16
17     listenfd = Open_listenfd(argv[1]);
18     while (1) {
19         clientlen = sizeof(struct sockaddr_storage);
20         connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
21         Getnameinfo((SA *) &clientaddr, clientlen, client_hostname, MAXLINE,
22                     client_port, MAXLINE, 0);
23         printf("Connected to (%s, %s)\n", client_hostname, client_port);
24         echo(connfd);
25         Close(connfd);
26     }
27     exit(0);
28 }
```

code/netp/echoserveri.c

图 11-21　迭代 echo 服务器的主程序

注意，简单的 echo 服务器一次只能处理一个客户端。这种类型的服务器一次一个地在客户端间迭代，称为*迭代服务器*(iterative server)。在第 12 章中，我们将学习如何建立更加复杂的*并发服务器*(concurrent server)，它能够同时处理多个客户端。

最后，图 11-22 展示了 echo 程序的代码，该程序反复读写文本行，直到 rio_readlineb 函数在第 10 行遇到 EOF。

code/netp/echo.c

```
1  #include "csapp.h"
2
3  void echo(int connfd)
4  {
5      size_t n;
6      char buf[MAXLINE];
7      rio_t rio;
8
9      Rio_readinitb(&rio, connfd);
10     while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
11         printf("server received %d bytes\n", (int)n);
12         Rio_writen(connfd, buf, n);
13     }
14 }
```

code/netp/echo.c

图 11-22　读和回送文本行的 echo 函数