的用户解释这个错误。

code/netp/tiny/tiny.c

```
1  void clienterror(int fd, char *cause, char *errnum,
2                   char *shortmsg, char *longmsg)
3  {
4      char buf[MAXLINE], body[MAXBUF];
5
6      /* Build the HTTP response body */
7      sprintf(body, "<html><title>Tiny Error</title>");
8      sprintf(body, "%s<body bgcolor=""ffffff"">\r\n", body);
9      sprintf(body, "%s%s: %s\r\n", body, errnum, shortmsg);
10     sprintf(body, "%s<p>%s: %s\r\n", body, longmsg, cause);
11     sprintf(body, "%s<hr><em>The Tiny Web server</em>\r\n", body);
12
13     /* Print the HTTP response */
14     sprintf(buf, "HTTP/1.0 %s %s\r\n", errnum, shortmsg);
15     Rio_writen(fd, buf, strlen(buf));
16     sprintf(buf, "Content-type: text/html\r\n");
17     Rio_writen(fd, buf, strlen(buf));
18     sprintf(buf, "Content-length: %d\r\n\r\n", (int)strlen(body));
19     Rio_writen(fd, buf, strlen(buf));
20     Rio_writen(fd, body, strlen(body));
21 }
```

code/netp/tiny/tiny.c

图 11-31　TINY clienterror 向客户端发送一个出错消息

回想一下，HTML 响应应该指明主体中内容的大小和类型。因此，我们选择创建 HTML 内容为一个字符串，这样一来我们可以简单地确定它的大小。还有，请注意我们为所有的输出使用的都是图 10-4 中健壮的 rio_writen 函数。

4. read_requesthdrs 函数

TINY 不使用请求报头中的任何信息。它仅仅调用图 11-32 中的 read_requesthdrs 函数来读取并忽略这些报头。注意，终止请求报头的空文本行是由回车和换行符对组成的，我们在第 6 行中检查它。

code/netp/tiny/tiny.c

```
1  void read_requesthdrs(rio_t *rp)
2  {
3      char buf[MAXLINE];
4
5      Rio_readlineb(rp, buf, MAXLINE);
6      while(strcmp(buf, "\r\n")) {
7          Rio_readlineb(rp, buf, MAXLINE);
8          printf("%s", buf);
9      }
10     return;
11 }
```

code/netp/tiny/tiny.c

图 11-32　TINY read_requesthdrs 读取并忽略请求报头