本行被截断，并用一个 NULL 字符结束。

rio_readnb 函数从文件 rp 最多读 n 个字节到内存位置 usrbuf。对同一描述符，对 rio_readlineb 和 rio_readnb 的调用可以任意交叉进行。然而，对这些带缓冲的函数的调用却不应和无缓冲的 rio_readn 函数交叉使用。

在本书剩下的部分中将给出大量的 RIO 函数的示例。图 10-5 展示了如何使用 RIO 函数来一次一行地从标准输入复制一个文本文件到标准输出。

code/io/cpfile.c

```
#include "csapp.h"

int main(int argc, char **argv)
{
    int n;
    rio_t rio;
    char buf[MAXLINE];

    Rio_readinitb(&rio, STDIN_FILENO);
    while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0)
        Rio_writen(STDOUT_FILENO, buf, n);
}
```

code/io/cpfile.c

图 10-5　从标准输入复制一个文本文件到标准输出

图 10-6 展示了一个读缓冲区的格式，以及初始化它的 rio_readinitb 函数的代码。rio_readinitb 函数创建了一个空的读缓冲区，并且将一个打开的文件描述符和这个缓冲区联系起来。

code/include/csapp.h

```
#define RIO_BUFSIZE 8192
typedef struct {
    int rio_fd;                /* Descriptor for this internal buf */
    int rio_cnt;               /* Unread bytes in internal buf */
    char *rio_bufptr;          /* Next unread byte in internal buf */
    char rio_buf[RIO_BUFSIZE]; /* Internal buffer */
} rio_t;
```

code/include/csapp.h

code/src/csapp.c

```
void rio_readinitb(rio_t *rp, int fd)
{
    rp->rio_fd = fd;
    rp->rio_cnt = 0;
    rp->rio_bufptr = rp->rio_buf;
}
```

code/src/csapp.c

图 10-6　一个类型为 rio_t 的读缓冲区和初始化它的 rio_readinitb 函数

RIO 读程序的核心是图 10-7 所示的 rio_read 函数。rio_read 函数是 Linux read 函数的带缓冲的版本。当调用 rio_read 要求读 n 个字节时，读缓冲区内有 rp->rio_cnt