准的。成员 d_name 是文件名，d_ino 是文件位置。

如果出错，则 readdir 返回 NULL，并设置 errno。可惜的是，唯一能区分错误和流结束情况的方法是检查自调用 readdir 以来 errno 是否被修改过。

```
#include <dirent.h>

int closedir(DIR *dirp);
```

返回：成功为 0；错误为 −1。

函数 closedir 关闭流并释放其所有的资源。图 10-11 展示了怎样用 readdir 来读取目录的内容。

code/io/readdir.c

```
1   #include "csapp.h"
2
3   int main(int argc, char **argv)
4   {
5       DIR *streamp;
6       struct dirent *dep;
7
8       streamp = Opendir(argv[1]);
9
10      errno = 0;
11      while ((dep = readdir(streamp)) != NULL) {
12          printf("Found file: %s\n", dep->d_name);
13      }
14      if (errno != 0)
15          unix_error("readdir error");
16
17      Closedir(streamp);
18      exit(0);
19  }
```

code/io/readdir.c

图 10-11　读取目录的内容

10.8　共享文件

可以用许多不同的方式来共享 Linux 文件。除非你很清楚内核是如何表示打开的文件，否则文件共享的概念相当难懂。内核用三个相关的数据结构来表示打开的文件：

- *描述符表*(descriptor table)。每个进程都有它独立的描述符表，它的表项是由进程打开的文件描述符来索引的。每个打开的描述符表项指向文件表中的一个表项。
- *文件表*(file table)。打开文件的集合是由一张文件表来表示的，所有的进程共享这张表。每个文件表的表项组成(针对我们的目的)包括当前的文件位置、引用计数(reference count)(即当前指向该表项的描述符表项数)，以及一个指向 v-node 表中对应表项的指针。关闭一个描述符会减少相应的文件表表项中的引用计数。内核不会删除这个文件表表项，直到它的引用计数为零。