

10.9 I/O 重定向

Linux shell 提供了 I/O 重定向操作符，允许用户将磁盘文件和标准输入输出联系起来。例如，键入

```
linux> ls > foo.txt
```

使得 shell 加载和执行 ls 程序，将标准输出重定向到磁盘文件 foo.txt。就如我们将在 11.5 节中看到的那样，当一个 Web 服务器代表客户端运行 CGI 程序时，它就执行一种相似类型的重定向。那么 I/O 重定向是如何工作的呢？一种方式是使用 dup2 函数。

```
#include <unistd.h>

int dup2(int oldfd, int newfd);
```

返回：若成功则为非负的描述符，若出错则为 -1。

dup2 函数复制描述符表表项 oldfd 到描述符表表项 newfd，覆盖描述符表表项 newfd 以前的内容。如果 newfd 已经打开了，dup2 会在复制 oldfd 之前关闭 newfd。

假设在调用 dup2(4,1) 之前，我们的状态如图 10-12 所示，其中描述符 1 (标准输出) 对应于文件 A (比如一个终端)，描述符 4 对应于文件 B (比如一个磁盘文件)。A 和 B 的引用计数都等于 1。图 10-15 显示了调用 dup2(4,1) 之后的情况。两个描述符现在都指向文件 B；文件 A 已经被关闭了，并且它的文件表和 v-node 表表项也已经被删除了；文件 B 的引用计数已经增加了。从此以后，任何写到标准输出的数据都被重定向到文件 B。

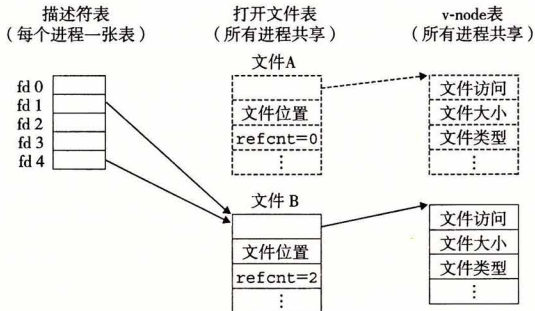




图 10-15 通过调用 dup2(4,1) 重定向到标准输出之后的内核数据结构。初始状态如图 10-12 所示

旁注 左边和右边的 hoinkies

为了避免和其他括号类型操作符比如 “]” 和 “[” 相混淆，我们总是将 shell 的 “>” 操作符称为 “右 hoinky”，而将 “<” 操作符称为 “左 hoinky”。

 **练习题 10.4** 如何用 dup2 将标准输入重定向到描述符 5？

 **练习题 10.5** 假设磁盘文件 foobar.txt 由 6 个 ASCII 码字符 “foobar” 组成，那么下列程序的输出是什么？

```
1 #include "csapp.h"
2
3 int main()
```