

## 实验六：存储管理

7. 编写一个程序，利用内存映象文件，实现 less 工具的功能（多屏显示）。

答：

程序代码如下：

```
1  #include <stdio.h>
2  #include <sys/stat.h>
3  #include <fcntl.h>
4  #include <unistd.h>
5  #include <sys/mman.h>
6  #include <string.h>
7  #include <memory.h>
8  #include <stdlib.h>
9  #include <stdio.h>
10 int lastrow(char *s, int d);
11 int nextrow(char *s, int d);
12 int onepage(char *s, int d);
13 int main()
14 {
15     int fd, play = 0;
16     char lab;
17     char *start;
18     struct stat sb;
19     fd = open("7.c", O_RDONLY); // 以只读方式打开文件
20     fstat(fd, &sb); // 获取文件的大小
21     start = mmap(NULL, sb.st_size, PROT_READ,
22     MAP_PRIVATE, fd, 0);
23     if (start == MAP_FAILED) // MAP_FAILED 表示映射失败
24         return (1);
25     play = onepage(start, play) + 1;
26     lab = getchar();
27     while (lab != 'q') // 输入的字符为 q，退出
28     {
29         if (play > sb.st_size) // 如果 onepage 返回的字节
30             // 数大于文件的大小，输入任意字符退出
31         }
```

```

30         lab = getchar();
31         break;
32     }
33     else if (lab == 'p')    // 输入 p, 继续读 10 行
34         play += onepage(start, play) + 1;
35     else if (lab == 'n')    // 输入 n, 显示下一行
36         play += nextrow(start, play) + 1;
37     else if (lab == 'l')    // 输入 l, 显示上一行
38         play = lastrow(start, play) + 1;
39     lab = getchar();
40 }
41 munmap(start, sb.st_size); // 解除映射
42 close(fd); // 关闭文件 fd
43 return 0;
44 }
45 int onepage(char *s, int d)
46 {
47     int i, count = 0; // count 在这里表示文件中行的数量
48     char *buffer = malloc(2048); // 配置内存空间, 由
49     // buffer 指向该空间
50     s += d; // 每 10 行作为一页输出
51     for (i = 0; i < 2048; i++)
52     {
53         if (s[i] == '\n')
54             count++;
55         if (count == 10)
56             break;
57     }
58     memcpy(buffer, s, i); // 从 s 处开始的地方拷贝 i 个字
59     // 节到 buffer
60     buffer[i] = '\0'; // 添加结束标识
61     printf("%s\n", buffer);
62     return i;
63 }
64 int nextrow(char *s, int d) // 下一行
65 {
66     int i;

```

```

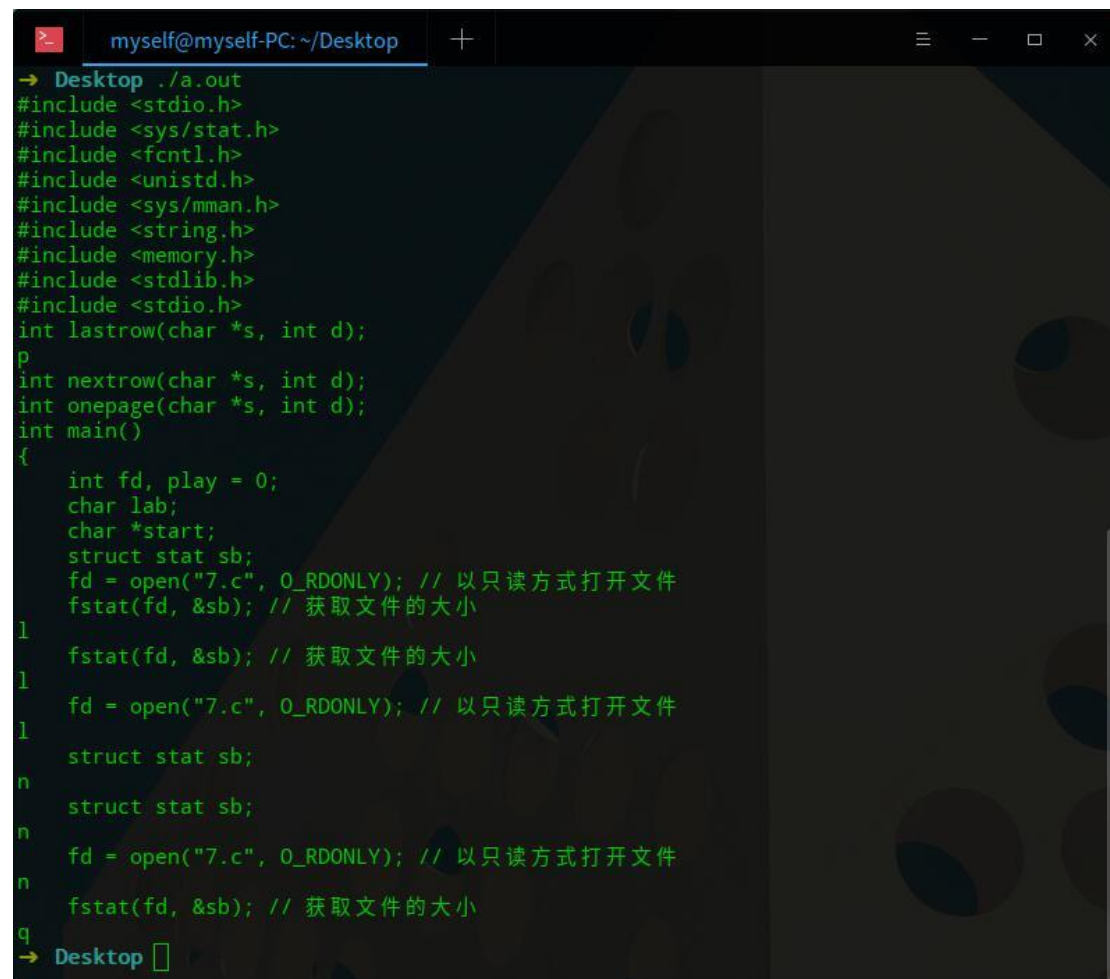
66     char *buffer = malloc(100);
67     s += d;
68     for (i = 0; i < 100; i++)
69         if (s[i] == '\n')
70             break;
71     memcpy(buffer, s, i);
72     buffer[i] = '\0';
73     printf("%s\n", buffer);
74     return i;
75 }
76 int lastrow(char *s, int d) // 上一行
77 {
78     int i, count = 0;
79     char *buffer = malloc(100);
80     int py = d;
81     for (; d > 0; d--)
82     {
83         if (s[d] == '\n')
84             count++;
85         if (count == 2)
86             break;
87     }
88     memcpy(buffer, s + d + 1, py - d - 2);
89     buffer[py - d - 2] = '\0';
90     printf("%s\n", buffer);
91     return d;
92 }

```

这段代码先使用 `fstat` 函数获得文件的大小，保证后续对文件内容的读取操作不发生越界，然后使用 `mmap` 函数将文件的内容映射到内存中。其中第一个参数 `start` 为 `NULL` 时表示由系统决定映射区的起始地址；第二个参数 `length` 表示映射区的长度，不足一页按一页处理，这里即为前面取得的文件大小；第三个参数期望的内存保护标志 `prot` 的 `PROT_READ` 表示页内容可以被读取；第四个参数映射的对象类型 `flags` 的 `MAP_PRIVATE` 表示建立一个写入时拷贝的私有映射。内存区域的写入不会影响到原文件；第五个参数 `fd` 表示有效的文件描述词，一般是由 `open` 函数返回；第六个参数 `off_toffset` 表示被映射对象内容的起点。`mmap` 函数的返回值为映射区内存的起始地址，此后调用的三个函数 `onepage`、`nextrow`、`lastrow` 的功能分别是显示一页内容、显示下一行内容、显示上一行内容。

运行该程序，映射 `7.c` 的文件内容到内存，使用 `p` 显示一页（10 行）内容，使用 `n` 显示

下一行内容，使用 l 显示上一行内容，效果如下图：



```
myself@myself-PC: ~/Desktop
→ Desktop ./a.out
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <string.h>
#include <memory.h>
#include <stdlib.h>
#include <stdio.h>
int lastrow(char *s, int d);
p
int nextrow(char *s, int d);
int onepage(char *s, int d);
int main()
{
    int fd, play = 0;
    char lab;
    char *start;
    struct stat sb;
    fd = open("7.c", O_RDONLY); // 以只读方式打开文件
    fstat(fd, &sb); // 获取文件的大小
1  fstat(fd, &sb); // 获取文件的大小
1  fd = open("7.c", O_RDONLY); // 以只读方式打开文件
1  struct stat sb;
n  struct stat sb;
n  fd = open("7.c", O_RDONLY); // 以只读方式打开文件
n  fstat(fd, &sb); // 获取文件的大小
q
→ Desktop □
```