课程编号：B080000070

《操作系统》实验报告

姓　　名	薛　　旗	学　　号	2 0 1 5 5 3 6 2
班　　级	软信-1503	指　导　教　师	王学毅
实　验　名　称	《操作系统》实验		
开　设　学　期	2016-2017 第二学期		
开　设　时　间	第 11 周——第 18 周		
报　告　日　期	2017 年 7 月 3 日		
评　定　成　绩		评　定　人	
		评　定　日　期	2017 年 7 月 5 日

东北大学软件学院

实验一　进程的同步与互斥

一、实验目的

（1）通过编写程序实现进程同步和互斥，掌握有关进程（线程）同步与互斥的原理，以及解决进程（线程）同步和互斥的算法，从而进一步巩固进程（线程）同步和互斥等有关的内容。

（2）学习使用 Windows 或 Linux 中基本的同步对象，掌握相应的 API 函数。

（3）掌握进程（线程）的概念，进程（线程）的控制原语或系统调用的使用。

二、实验内容

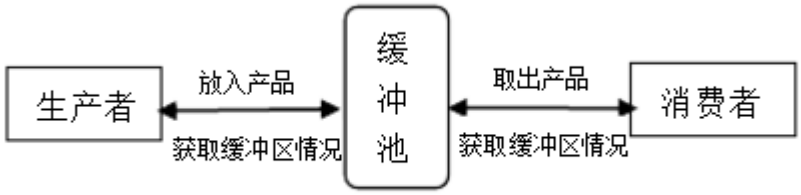
采用进程（线程）同步和互斥的技术编写程序实现生产者-消费者问题。

要求：

生产者进程（线程）生产信息，例如它可以是计算进程。消费者进程（线程）使用信息，它可以是输出打印进程。

三、程序设计分析

（1）由于生产者和消费者彼此独立，且运行速度不确定，所以很可能出现生产者已产生了信息而消费者却没有来得及接受信息这种情况。为此，需要引入由一个或者若干个存储单元组成的临时存储区，以便存放生产者所产生的信息，平滑进程间由于速度不确定所带来的问题。这个临时存储区叫做缓冲区，通常用一维数组来表示。由一个或若干个存储单元组成的缓冲区叫作“有穷缓冲区”。



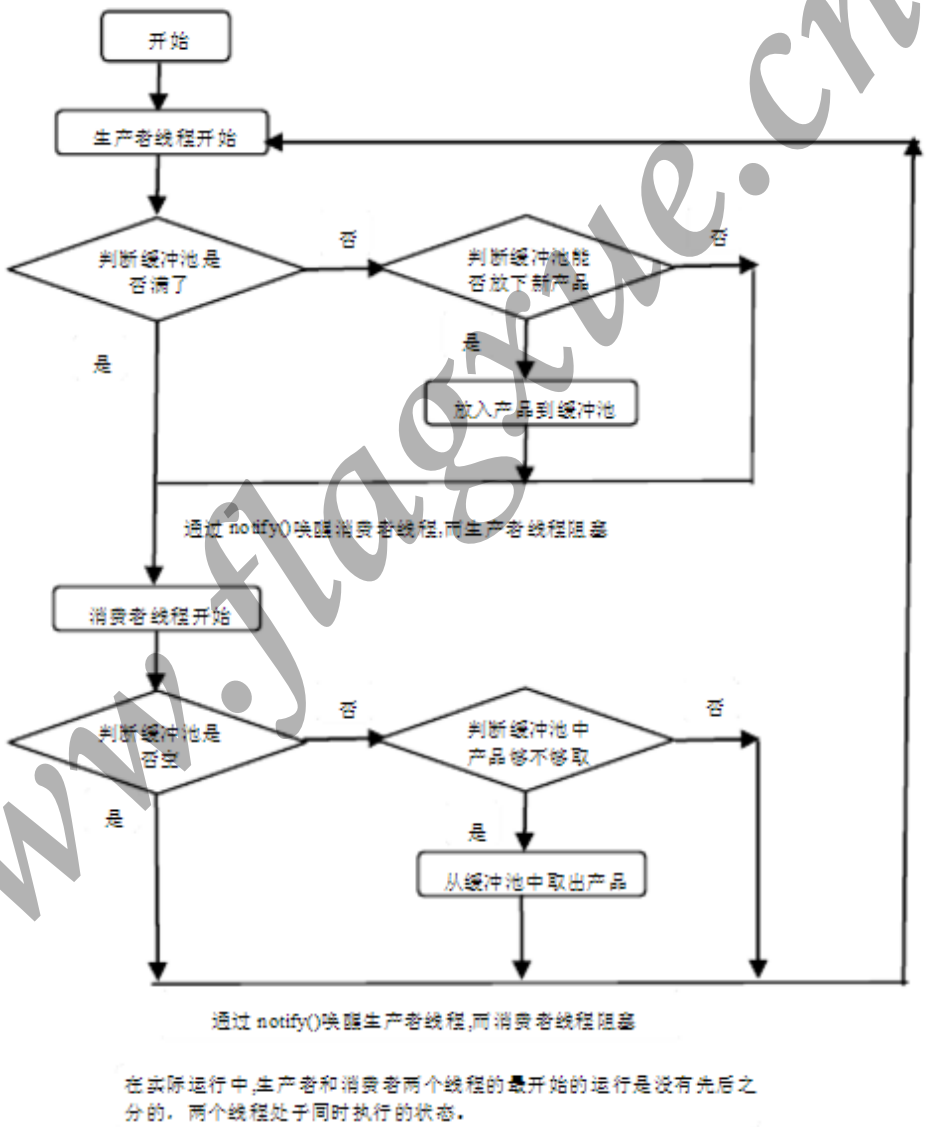
原理图

（2）如果想使生产者进程（线程）和消费者进程（线程）协调合作，则必须使它们遵循如下规则:

1）只要缓冲区有存储单元，生产者都可往其中存放信息；当缓冲区已满时，若任意生产者提出写要求，则都必须等待；

2）只要缓冲区中有消息可取，消费者都可从缓冲区中取出消息；当缓冲区为空时，若任意消费者想取出信息，则必须等待；

3）生产者们和消费者们不能同时读、写缓冲区。



流程图

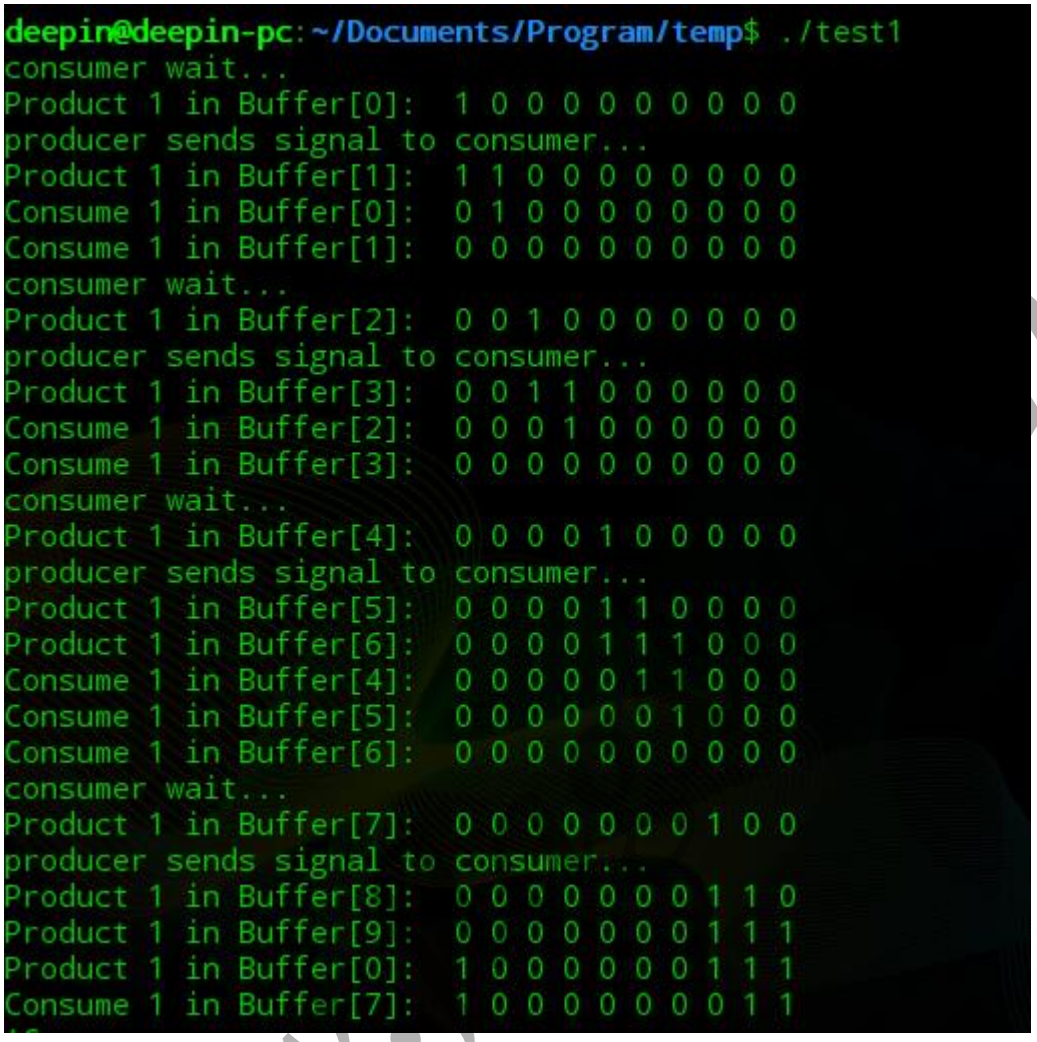
四、数据结构及符号说明

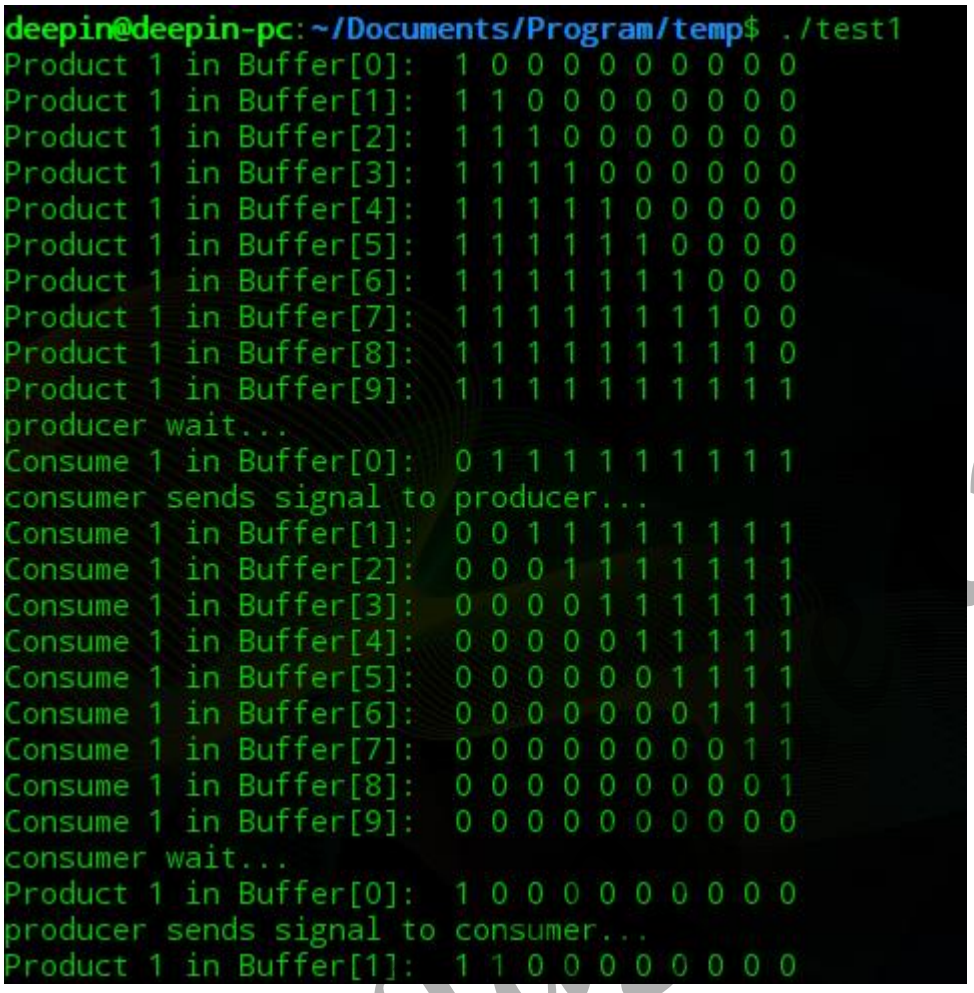
```
int Front = 0;           //队首指针，取出信息的位置(Buffer[Front])
int Rear = 0;            //队尾指针，当前信息应存放位置(Buffer[Rear])
int Buffer[n] = { 0 };   //初始化缓冲区为0
int current_size = n;    //当前缓冲区容量
int producer_sign = 0;   //生产者阻塞标志
int consumer_sign = 0;   //消费者阻塞标志

//生产者线程用于唤醒消费者线程的条件变量
pthread_cond_t cond_producter = PTHREAD_COND_INITIALIZER;
//消费者线程用于唤醒生产者线程的条件变量
pthread_cond_t cond_consumer = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex;   //互斥信号量
```

五、函数说明

	调用函数	说明
1	int main(void)	主函数
2	void producer(void)	生产者
3	void consumer(void)	消费者
4	void print(void)	打印缓冲区情况

六、实验数据与实验结果


deepin@deepin-pc:~/Documents/Program/temp$ ./test1
consumer wait...
Product 1 in Buffer[0]: 1 0 0 0 0 0 0 0 0 0
producer sends signal to consumer...
Product 1 in Buffer[1]: 1 1 0 0 0 0 0 0 0 0
Consume 1 in Buffer[0]: 0 1 0 0 0 0 0 0 0 0
Consume 1 in Buffer[1]: 0 0 0 0 0 0 0 0 0 0
consumer wait...
Product 1 in Buffer[2]: 0 0 1 0 0 0 0 0 0 0
producer sends signal to consumer...
Product 1 in Buffer[3]: 0 0 1 1 0 0 0 0 0 0
Consume 1 in Buffer[2]: 0 0 0 1 0 0 0 0 0 0
Consume 1 in Buffer[3]: 0 0 0 0 0 0 0 0 0 0
consumer wait...
Product 1 in Buffer[4]: 0 0 0 0 1 0 0 0 0 0
producer sends signal to consumer...
Product 1 in Buffer[5]: 0 0 0 0 1 1 0 0 0 0
Product 1 in Buffer[6]: 0 0 0 0 1 1 1 0 0 0
Consume 1 in Buffer[4]: 0 0 0 0 0 1 1 0 0 0
Consume 1 in Buffer[5]: 0 0 0 0 0 0 1 0 0 0
Consume 1 in Buffer[6]: 0 0 0 0 0 0 0 0 0 0
consumer wait...
Product 1 in Buffer[7]: 0 0 0 0 0 0 0 1 0 0
producer sends signal to consumer...
Product 1 in Buffer[8]: 0 0 0 0 0 0 0 1 1 0
Product 1 in Buffer[9]: 0 0 0 0 0 0 0 1 1 1
Product 1 in Buffer[0]: 1 0 0 0 0 0 0 1 1 1
Consume 1 in Buffer[7]: 1 0 0 0 0 0 0 0 1 1



七、实验总结

通过本次实验，我对进程的同步与互斥有了更深的理解，对生产者消费者模型有了更深入的认识。

八、思考题

（1）如何控制进程间的相互通信？

答：当相互合作的进程处于同一计算机系统时，通常在它们之前是采用直接通信方式，即由源进程利用发送命令直接将消息(message)挂到目标进程的消息队列上，以后由目标进程利用接收命令从其消息队列中取出信息。当相互合作的进程处于不同的系统中时，常采用间接通信方式，即由源进程利用发送命令将消息送入中间实体——邮箱，以后由目标进程从中取走消息。

（2）什么是进程的同步？什么是进程的互斥？分别有哪些实现方式？

答：进程的同步：并发进程在一些关键点上可能需要互相等待与互通消息，这种相互制约的等待与互通消息称为进程同步。

进程的互斥：当多个进程需要使用相同的资源，而此类资源在任意时刻却只能供一个进程使用，获得资源的进程可以继续执行，没有获得资源的进程必须等待，这种对共享资源的排他性使用造成的进程间的间接制约关系称为进程的互斥。

实现方法：临界区，互斥量，信号量，事件。

六、实验用程序清单

```
#include <stdio.h>          // printf()
#include <semaphore.h>      // sem_init()
#include <stdlib.h>         // exit()
#include <pthread.h>        // pthread_create(), pthread_join()
#define n 10 //缓冲区大小
#define NUM 5//消费者或生产者数目

int Front = 0;    //队首指针，取出信息的位置(Buffer[Front])
int Rear = 0;     //队尾指针，当前信息应存放位置(Buffer[Rear])
int Buffer[n] = { 0 };//初始化缓冲区为0
int current_size = n; //当前缓冲区容量
int producer_sign = 0;//生产者阻塞标志
int consumer_sign = 0;//消费者阻塞标志

//生产者线程用于唤醒消费者线程的条件变量
pthread_cond_t cond_producter = PTHREAD_COND_INITIALIZER;
//消费者线程用于唤醒生产者线程的条件变量
pthread_cond_t cond_consumer = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex;//互斥信号量

void print(void) {    //打印缓冲区情况
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", Buffer[i]);
    }
    printf("\n");
}

void producer(void) { //生产者
```

```
30      int product = 1;  //产品
31      while (1) {
32          pthread_mutex_lock(&mutex);     //锁住互斥量
33          while (current_size <= 0) {//空间不足，禁止生产者生产产品，生产者线程阻塞
34              producer_sign = 1;          //设置生产者线程阻塞标志
35              printf("producer wait...\n");
36              pthread_cond_wait(&cond_producter, &mutex);
37          }
38          printf("Product %d in Buffer[%d]:  ", product, Rear);
39          Buffer[Rear] = 1;
40          print();
41          sleep(1);
42          Rear = (Rear + 1) % n;
43          current_size--;
44          pthread_mutex_unlock(&mutex);
45          if (consumer_sign == 1) {
46              printf("producer sends signal to consumer...\n");
47              pthread_cond_signal(&cond_consumer);
48              consumer_sign = 0;     //解除消费者阻塞标志
49          }
50      }
51  }
52
53  void consumer(void) { //消费者
54      int product = 1;  //产品
55      while (1) {
56          pthread_mutex_lock(&mutex);
57          while (current_size >= n) {    //没有产品，禁止消费者消费，消费者线程阻塞
58              consumer_sign = 1;          //设置消费者线程阻塞标志
59              printf("consumer wait...\n");
60              pthread_cond_wait(&cond_consumer, &mutex);
61          }
62          //printf("maintain_size:  %d\n", current_size);
63          printf("Consume %d in Buffer[%d]:  ", product, Front);
64          Buffer[Front] = 0;
65          print();
66          sleep(1);
67          Front = (Front + 1) % n;
68          current_size++;
69          pthread_mutex_unlock(&mutex);
70          if (producer_sign == 1) {
71              printf("consumer sends signal to producer...\n");
72              pthread_cond_signal(&cond_producter);
73              producer_sign = 0;     //解除生产者阻塞标志
```

```
74          }
75      }
76  }
77
78  int main(void) {
79      pthread_t t1, t2;
80      pthread_mutex_init(&mutex, NULL);  //初始化互斥信号量
81      pthread_create(&t1, NULL, (void *)producer, NULL);
82      pthread_create(&t2, NULL, (void *)consumer, NULL);
83      pthread_join(&t1, NULL);
84      pthread_join(&t2, NULL);
85      exit(0);
86  }
```

实验二　处理机调度

一、实验目的

（1）加深对处理机调度的作用和工作原理的理解。

（2）进一步认识并发执行的实质。

二、实验内容

设计一个按时间片轮转法实现处理器调度的程序。

为每个进程任意确定一组“要求运行时间”，启动所设计的处理器调度程序，显示或打印逐次被选中的进程名以及进程控制块的动态变化过程。

三、程序设计分析

（1）每次运行所设计的处理器调度程序前，为每个进程任意确定它的“要求运行时间”。

（2）将进程按顺序排成循环队列，用指针指出队列连接情况。另用一标志单元记录轮到运行的进程。

（3）进程运行一次后，应把该进程的进程控制块中的指针值送到标志单元，以指示下一个轮到运行的进程。同时，应判断该进程的要求运行时间与已运行时间，若该进程的要求运行时间≠已运行时间，则表示它尚未执行结束，应待到下一轮时再运行。若该进程的要求运行时间=已运行时间，则表示它已经执行结束，应指导它的状态修改成“结束”（E）且退出队列。此时，应把该进程的进程控制块中的指针值送到前面一个进程的指针位置。

（4）若“就绪”状态的进程队列不为空，则重复执行，直到所有的进程都成为“结束”状态。

四、数据结构及符号说明

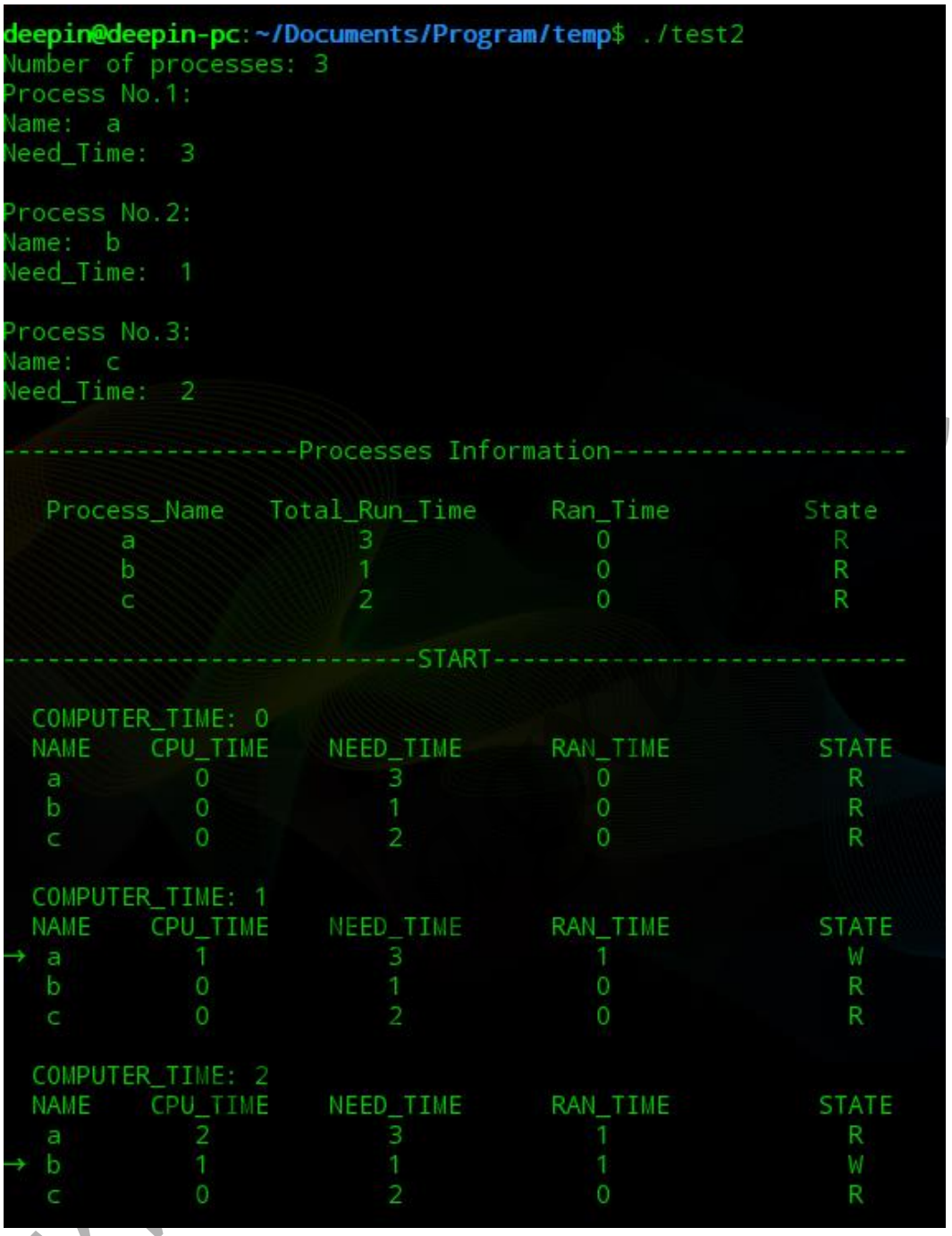
```
typedef struct PCB {
    char progress_name[10];
    struct PCB *next;
    int needTime;
    int ranTime;
    char status;
}PCB, *PPCB; //指向该 PCB 的指针
```

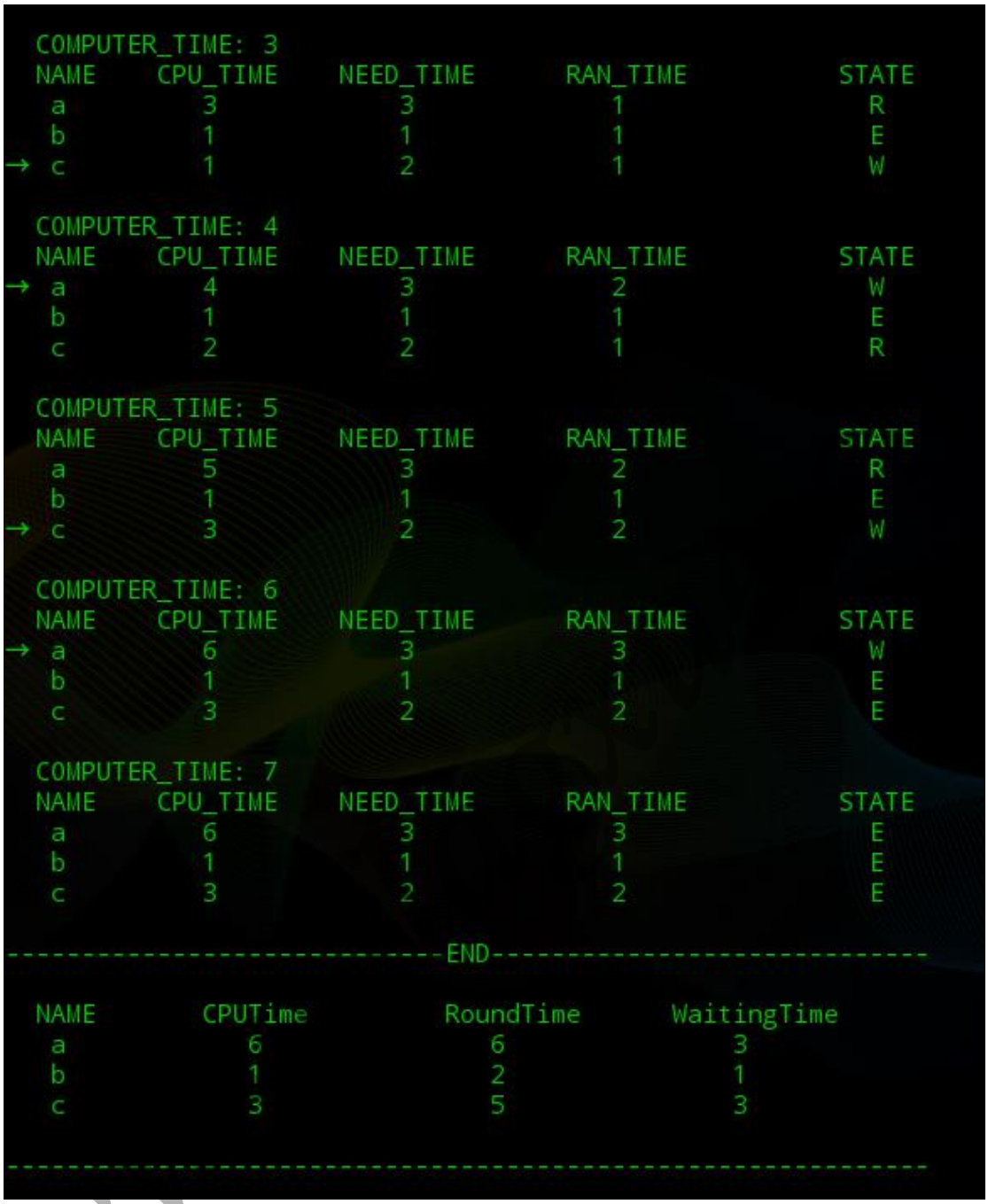
```
typedef struct List {
    int info;
    int cpuTime;
    int turnAroundTime;
    int waitingTime;
    struct List *next;
} List, *pList;   //进程信息
```

五、函数说明

	调用函数	说明
1	int main(void)	主函数
2	void initPCB(PPCB, pList)	创建链表
3	void initDisplay(PPCB)	打印初始化链表后信息
4	void RR(PPCB, pList)	时间片轮转算法
5	void resultDisplay(PPCB, pList, int)	打印每轮执行结果
6	void endDisplay(PPCB, pList, int)	打印统计信息

六、实验数据与实验结果(见下页)


deepin@deepin-pc:~/Documents/Program/temp$ ./test2
Number of processes: 3
Process No.1:
Name: a
Need_Time: 3
Process No.2:
Name: b
Need_Time: 1
Process No.3:
Name: c
Need_Time: 2
--------------------Processes Information--------------------
Process_Name Total_Run_Time Ran_Time State
a 3 0 R
b 1 0 R
c 2 0 R
----------------------------START-----------------------------
COMPUTER_TIME: 0
NAME CPU_TIME NEED_TIME RAN_TIME STATE
a 0 3 0 R
b 0 1 0 R
c 0 2 0 R
COMPUTER_TIME: 1
NAME CPU_TIME NEED_TIME RAN_TIME STATE
→ a 1 3 1 W
b 0 1 0 R
c 0 2 0 R
COMPUTER_TIME: 2
NAME CPU_TIME NEED_TIME RAN_TIME STATE
a 2 3 1 R
→ b 1 1 1 W
c 0 2 0 R



七、实验总结

通过编写并运行这个实验，对时间片调度算法有了更深刻的了解，并且在学习过程中显著提高了编程能力，获益匪浅。

八、思考题

（1）处理机调度的目的？

答：处理机调度就是一个按照一定的算法来取作业的过程，并为选取的作业分配资

源，建立相应的进程，以及作业结束后的一系列处理工作。这些工作量的最主要目的还是为了提高效率，能最快完成作业。但可行的算法是不尽相同的，因为不同的算法也是为满足统统的需求。

（2）你实现优先权调度算法的思想？

答：当执行序列不为空时，进行一系列的操作（即 cputime 加一，needtime 减一，优先级减三），此时执行状态为“R”，（附：等待状态为“W”）如果执行到 runtime 为零，则结束该进程，并进程的状态转变为“F”。同时还有一个 ready，主要是讲进程调入执行。至于执行哪一个进程，主要是根据设置优先级的高低决定，通过判断，优先级高的先进入执行。因此可以看出，主要就是优先级的比较，高的执行，低的等待。

（3）你采用时间片轮转法实现处理机调度的思想？

答：为个进程被分配一个时间段，如果在时间片结束时进程还在运行，则 CPU 将被剥夺并分配给另一个进程。如果进程在时间片结束前阻塞或结束，则 CPU 当即进行切换。调度程序所要做的就是维护一张就绪进程列表，当进程用完它的时间片后，它被移到队列的末尾。

（4）比较效率如何？

答：RR 算法的性能很大程度上依赖于时间片的大小。在极端情况下，如果时间片非常大，那么 RR 算法与 FCFS 算法一样。如果时间片很小，那么 RR 算法称为处理器共享。

九、实验用程序清单

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>   //sleep()
5
6  typedef struct PCB {
7      char progress_name[10];
8      struct PCB *next;
9      int needTime;
10     int ranTime;
11     char status;
12 }PCB, *PPCB; //指向该PCB的指针
13
14 typedef struct List {
15     int info;
```

```
    int cpuTime;
    int turnAroundTime;
    int waitingTime;
    struct List *next;
} List, *pList;   //进程信息
int ProNum;

void initPCB(PPCB pHead, pList L) {//尾插法创建链表
    int i;
    PCB *s, *r;  //s:z指向新生成的结点 r：始终指向终端结点
    List *s2, *r2;
    printf("Number of processes: ");
    scanf("%d", &ProNum);
    r = pHead;   //r指向头结点,此时头结点也是终端结点
    r2 = L;
    for (i = 0; i < ProNum; i++) {
        s = (PPCB)malloc(sizeof(PCB)); //s指向新申请的结点
        s2 = (List *)malloc(sizeof(List));
        printf("Process No.%d:\n", i + 1);
        printf("Name:  ");
        scanf("%s", s->progress_name);
        printf("Need_Time:  ");
        scanf("%d", &s->needTime);
        s->ranTime = 0;
        s->status = 'R';
        s2->info = 0;
        s2->cpuTime = 0;
        s2->turnAroundTime = 0;
        s2->waitingTime = 0;
        printf("\n");
        r->next = s;
        r2->next = s2;
        r = s;   //r始终指向终端结点
        r2 = s2;
    }
    r->next = pHead->next;//链表建立完成
    r2->next = L->next;
}

void initDisplay(PPCB pHead) {
    int i;
    PCB *p = pHead->next;
    printf("--------------------Processes Information--------------------\n\n");
    printf("   Process_Name    Total_Run_Time     Ran_Time         State\n");
```

```
60      for (i = 0; i < ProNum; i++) {
61          printf("\t%s\t\t%d\t\t%d\t\t%c", p->progress_name, p->needTime, p->ranTime,
    p->status);
62          printf("\n");
63          p = p->next;
64      }
65  }
66
67  void resultDisplay(PPCB pHead, pList L, int count) {
68      int i;
69      PPCB p = pHead->next;
70      pList p2 = L->next;
71      printf("  COMPUTER_TIME: %d\n", count);
72      printf("  NAME    CPU_TIME    NEED_TIME      RAN_TIME          STATE\n");
73      for (i = 0; i < ProNum; i++) {
74          if (p->status == 'W') {
75              printf("→  %s\t     %d\t\t  %d\t\t%d\t\t %c\n", p->progress_name,
    p2->cpuTime, p->needTime, p->ranTime, p->status);
76              p->status = 'R';
77          }
78          else {
79              printf("   %s\t     %d\t\t  %d\t\t%d\t\t %c\n", p->progress_name,
    p2->cpuTime, p->needTime, p->ranTime, p->status);
80          }
81          p = p->next;
82          p2 = p2->next;
83      }
84      printf("\n");
85      sleep(1);     //逐条打印，模拟时间片轮转
86  }
87
88  void endDisplay(PPCB pHead, pList L, int count) {
89      int i;
90      PPCB p = pHead->next;
91      pList p2 = L->next;
92      printf("  NAME       CPUTime          RoundTime       WaitingTime\n");
93      for (i = 0; i < ProNum; i++) {
94          printf("   %s\t\t%d\t\t%d\t\t%d\n", p->progress_name, p2->cpuTime,
    p2->turnAroundTime, p2->waitingTime);
95          p = p->next;
96          p2 = p2->next;
97      }
98      printf("\n");
99  }
```

```
100
101 void RR(PPCB pHead, pList L) {
102     printf("\n--------------------------START--------------------------\n\n");
103     int i;
104     int flag = ProNum;
105     int count = 0;    //记录轮转数
106     PPCB p = pHead->next; //p为标记单元
107     PPCB temp = pHead->next;
108     pList p2 = L->next;   //p2为进程信息标记单元
109     pList tempP = L->next;
110     resultDisplay(pHead, L, count);
111     while (p->needTime > p->ranTime) {
112         count++;
113         p->ranTime++;
114         p->status = 'W';
115         p2->info = 1;
116         for (i = 0; i< ProNum; i++) {
117             if (tempP->info == 1 && temp->status != 'E')
118                 tempP->cpuTime++;
119             tempP = tempP->next;
120             temp = temp->next;
121         }
122
123         resultDisplay(pHead, L, count);
124
125         if (p->needTime == p->ranTime) {
126             p->status = 'E';
127             flag--;
128             p2->turnAroundTime = count;    //周转时间
129             p2->waitingTime = count - p->needTime;  //等待时间
130         }
131         p = p->next;
132         p2 = p2->next;
133
134         while (flag && p->needTime == p->ranTime) {
135             p = p->next;
136             p2 = p2->next;
137         }
138     }
139     count++;
140     resultDisplay(pHead, L, count++);
141     printf("--------------------------END--------------------------\n\n");
142     endDisplay(pHead, L, count++);
143     printf("-----------------------------------------------------------\n\n");
```

```
}

int main(void) {
    PPCB pHead;
    pList L;
    pHead = (PPCB)malloc(sizeof(PCB)); //建立头结点
    L = (pList)malloc(sizeof(List));
    initPCB(pHead, L);
    initDisplay(pHead);
    RR(pHead, L);
    return 0;
}
```

实验三　存储管理

一、实验目的

（1）加深对存储管理的作用和工作原理的理解。

（2）进一步认识主存空间的分配和回收方法。

（3）进一步认识虚拟存储器的工作原理。

二、实验内容

第一题：模拟分页式存储管理中硬件的地址转换和产生缺页中断。

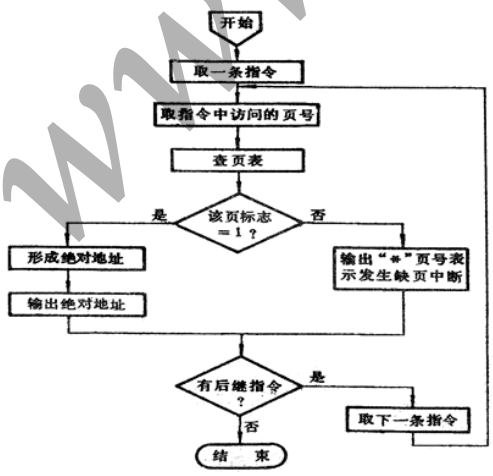
设计一个“地址转换”程序来模拟硬件的地址转换工作。当访问的页在主存时，则形成绝对地址，但不去模拟指令的执行，而用输出转换后的地址来代替一条指令的执行。

第二题：用先进先出（FIFO）页面调度算法处理缺页中断。

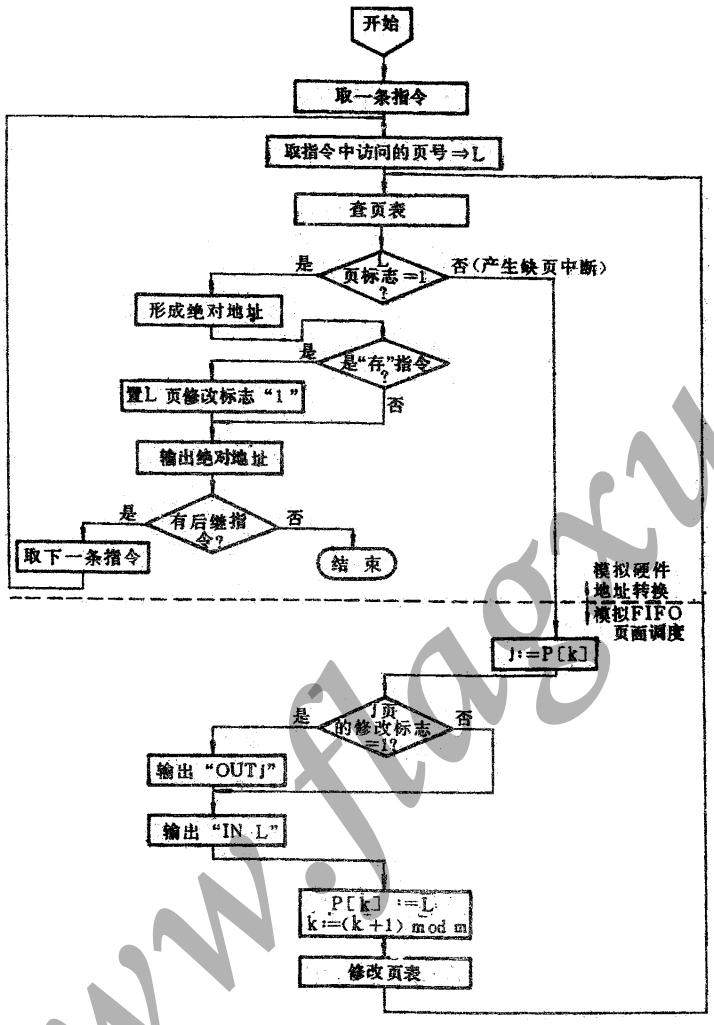
运行你所设计的程序，显示或打印每次调出和装入的页号，以及执行了最后一条指令后的数组 P 的值。

三、程序设计分析

(1)地址转换模拟算法



(2)FIFO 页面调度模拟算法



四、数据结构及符号说明

第一题：

```
typedef struct {
    int pagenum; //页号
    int flag;        //标志
    int blocknum;//主存块号
    int location;//磁盘位置
}table;  //页表
```

```
table p[7] = {
    { 0, 1, 5, 11 },
    { 1, 1, 8, 12 },
    { 2, 1, 9, 13 },
    { 3, 1, 1, 21 },
    { 4, 0, 0, 22 },
    { 5, 0, 0, 23 },
    { 6, 0, 0, 121 }
};//初始化作业页表
```

第二题：

```
typedef struct {
    int pagenum; //页号
    int flag;         //标志
    int blocknum;//主存块号
    int modify;       //修改标志
    int location;//磁盘位置
}table;  //页表
```

```
table p1[7] = {   //初始化页表
    { 0, 1, 5, 0, 11 },
    { 1, 1, 8, 0, 12 },
    { 2, 1, 9, 0, 13 },
    { 3, 1, 1, 0, 21 },
    { 4, 0, 0, 0, 22 },
    { 5, 0, 0, 0, 23 },
    { 6, 0, 0, 0, 121 }
};
```

```
typedef struct instru_list {
    char operation;
    int pagenum;
    int page_address;
}operate;    //指令序列
```

```
operate p2[12] = {    //初始化指令序列
    { '+', 0, 70 },
    { '+', 1, 50 },
    { '*', 2, 15 },
    { 'S', 3, 21 },    //存
    { 'T', 0, 56 },    //取
    { '-', 6, 40 },
    { 'M', 4, 53 },    //移
    { '+', 5, 23 },
    { 'S', 1, 37 },
    { 'T', 2, 78 },
    { '+', 4, 01 },
    { 'S', 6, 84 }
};
```

六、实验数据与实验结果

第一题：

```
deepin@deepin-pc:~/Documents/Program/temp$ ./test3_1

---Input (-1 -1) to exit---

Please input page number and page address: 1 30
page_address: 30        absolute_address: 1054

Please input page number and page address: 2 70
page_address: 70        absolute_address: 1222

Please input page number and page address: 4 20
*4 Page Fault

Please input page number and page address: 5 10
*5 Page Fault

Please input page number and page address: 3 100
page_address: 100       absolute_address: 228

Please input page number and page address: -1 -1
```

第二题：

```
deepin@deepin-pc:~/Documents/Program/temp$ ./test3_2

------------------Initial Table------------------

PageNum     Flag     BlockNum     Modify     Location
   0          1          5          0          11
   1          1          8          0          12
   2          1          9          0          13
   3          1          1          0          21
   4          0          0          0          22
   5          0          0          0          23
   6          0          0          0          121

---------Instruction Sequence--------

Operation     PageNum     Page_address
   +             0            70
   +             1            50
   *             2            15
   S             3            21
   T             0            56
   -             6            40
   M             4            53
   +             5            23
   S             1            37
   T             2            78
   +             4            1
   S             6            84
```

```
-----------------------------------------------------Outcome-----------------------------------------------------

Operation   PageNum   Page_addr   Abso_addr   Flag   BlockNum   Modify   Location   Abso_addr   Modify      Remark
   +          0         70          710        1        5                  11         710         0        INITIA
   +          1         50          1074       1        8                  12         1074        0        INITIA
   *          2         15          1167       1        9                  13         1167        0        INITIA
   S          3         21          149        1        1                  21         149         0        INITIA
   T          0         56          696        1        5                  11         696         0         In 0
   -          6         40          *6         0        5        6->0      121        680         0         In 6
   M          4         53          *4         0        8        4->1      22         1077        0         In 4
   +          5         23          *5         0        9        5->2      23         1175        0         In 5
   S          1         37          *1         0        1        1->3      12         165         1      Out 3, In 1
   T          2         78          *2         0        5        2->6      13         718         0         In 2
   +          4         1           1025       1        8                  22         1025        0         In 4
   S          6         84          *6         0        8        6->4      121        1108        1         In 6
```

七、实验总结

通过本实验，我加深对存储管理的作用和工作原理的理解，并进一步认识主存空间的分配和回收方法。获益匪浅

八、思考题

（1）先进先出页面调度算法的思想？

答： 每当有新的页面进入时将会产生页面中断，而被淘汰的将是最先进入的页面。按照各个作业进入系统的自然次序来调度作业。这种调度算法的优点是实现简单，公平。其缺点是没有考虑到系统中各种资源的综合使用情况，往往使短作业的用户不满意，因为短作业等待处理的时间可能比实际运行时间长得多。

（2）最近最少用（LRU）页面调度算法思想？

答：每次调换出的页面是所有内存页面中最迟将被使用的。该算法可以用寄存器组和栈实现，性能较好。

（3）比较两种调度算法的效率（哪种调度算法使产生缺页中断的次数少）？

答：最近最少用(LRU)算法产生缺页中断的次数少，算法效率更高。

FIFO 按照先进先出的原理淘汰数据，正好符合队列的特性，数据结构上使用队列 Queue 来实现。

LRU（算法根据数据的历史访问记录来进行淘汰数据，其核心思想是“如果数据最近被访问过，那么将来被访问的几率也更高”。

九、实验用程序清单

第一题：

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>

#define block_size 128

typedef struct {
    int pagenum; //页号
    int flag;        //标志
    int blocknum;//主存块号
    int location;//磁盘位置
}table;  //页表

table p[7] = {
    { 0, 1, 5, 11 },
    { 1, 1, 8, 12 },
```

```
    { 2, 1, 9, 13 },
    { 3, 1, 1, 21 },
    { 4, 0, 0, 22 },
    { 5, 0, 0, 23 },
    { 6, 0, 0, 121 }
};//初始化作业页表

int main(void) {
    int page;
    int offset;
    int memaddress;
    printf("\n---Input (-1 -1) to exit---\n\n");

    while (1) {
        setbuf(stdin, NULL);
        printf("Please input page number and page address: ");
        scanf("%d %d", &page, &offset);
        if (page == -1 && offset == -1) {
            printf("\n");
            exit(1);
        }
        else if (page < 0 || page > 6 || offset < 0) {
            printf("Out of range! Please input again!\n\n");
        }
        else {

            if (p[page].flag != 0) {  //该页已经装入主存
                memaddress = p[page].blocknum * block_size + offset;
                printf("page_address: %d\tabsolute_address: %d\n\n", offset, memaddress);
            }
            else {
                printf("*%d Page Fault\n\n", page);
            }

        }
    }

    return 0;
}
```

第二题：

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
```

```
#include <stdlib.h>

#define block_size 128//块长
#define operation_num 12  //指令个数

typedef struct {
    int pagenum; //页号
    int flag;        //标志
    int blocknum;//主存块号
    int modify;      //修改标志
    int location;//磁盘位置
}table;  //页表

typedef struct instru_list {
    char operation;
    int pagenum;
    int page_address;
}operate;    //指令序列


table p1[7] = {  //初始化页表
    { 0, 1, 5, 0, 11 },
    { 1, 1, 8, 0, 12 },
    { 2, 1, 9, 0, 13 },
    { 3, 1, 1, 0, 21 },
    { 4, 0, 0, 0, 22 },
    { 5, 0, 0, 0, 23 },
    { 6, 0, 0, 0, 121 }
};

operate p2[12] = {    //初始化指令序列
    { '+', 0, 70 },
    { '+', 1, 50 },
    { '*', 2, 15 },
    { 'S', 3, 21 },   //存
    { 'T', 0, 56 },   //取
    { '-', 6, 40 },
    { 'M', 4, 53 },   //移
    { '+', 5, 23 },
    { 'S', 1, 37 },
    { 'T', 2, 78 },
    { '+', 4, 01 },
    { 'S', 6, 84 }
};
```

```
48
49
50
51  int main(void) {
52      int i, j, page, rep_page, modify, flag, memaddress;
53      int k = 0;
54      int m = 4;
55      int p[4] = { 0, 1, 2, 3 };
56
57      printf("\n------------------Initial Table------------------\n\n");
58      printf("PageNum    Flag    BlockNum    Modify    Location\n");
59      for (i = 0; i < 7; i++) {
60          printf("   %d\t    %d\t     %d\t        %d\t    %d\n", p1[i].pagenum, p1[i].flag, p1[i].blocknum, p1[i].modify, p1[i].location);
61      }
62      printf("\n---------Instruction Sequence--------\n\n");
63
64      printf("Operation    PageNum    Page_address\n");
65      for (i = 0; i < 12; i++) {
66          printf("   %c\t        %d\t    %d\n", p2[i].operation, p2[i].pagenum, p2[i].page_address);
67      }
68
69      printf("\n-------------------------------------------------------Outcome-------------------------
70      printf("Operation   PageNum   Page_addr   Abso_addr   Flag   BlockNum   Modify   Location   Abso_addr   Modify    Remark\n");
71
72      for (i = 0; i < operation_num; i++) {
73          page = p2[i].pagenum; //待处理的页号
74          flag = p1[page].flag; //待处理的页标志，判断该页是否已经装入主存
75          if (flag == 0) {  //该页未装入主存
76              rep_page = p[k];  //将被替换的页号
77              modify = p1[rep_page].modify;  //将被替换的页的修改标志
78              p1[page].blocknum = p1[rep_page].blocknum;  //装入主存
79              p1[page].flag = 1;    //该页装入主存，标志位改为1
80              p1[rep_page].flag = 0;//被替代的页标志位改为0
81              p1[rep_page].modify = 0;  //被替代的页未被修改
82              p1[rep_page].blocknum = -1;    //移出主存，无对应块号
83              p[k] = page;
84              k = (k + 1) % m;
85          }
86  //待处理页已经装入主存
87          memaddress = p1[page].blocknum * block_size + p2[i].page_address;
88          if (p2[i].operation == 'S')
89              p1[page].modify = 1;
90          printf("   %c\t      %d\t %d\t    ", p2[i].operation, p2[i].pagenum, p2[i].page_address);
91          if (flag == 1)    //待处理页已装入主存
```

```
92                  printf("%d      \t%d        %d      \t\t      %d      \t%d\t  ", memaddress, flag, p1[page].blocknum, p1[page].location, memaddress);
93          else
94  printf("*%d     \t%d        %d\t  %d->%d\t     %d      \t%d\t  ", p2[i].pagenum, flag, p1[page].blocknum, p2[i].pagenum, rep_page, p1[page].location, memaddress);
95
96          if (i < 4)
97              printf(" 0\t    INITIA");
98          else {
99              if (modify == 1)
100                 printf(" %d\t Out %d, In %d", p1[page].modify, rep_page, page);
101             else
102                 printf(" %d\t     In %d", p1[page].modify, page);
103         }
104         printf("\n");
105     }
106     printf("\n");
107
108     return 0;
109 }
110
```

实验四　文件系统

一、实验目的

（1）加深理解文件系统的内部功能及内部实现。

（2）进一步认识文件系统的作用。

二、实验内容

用高级语言编写和调试一个简单的文件系统，模拟文件管理的工作过程。

要求设计一个 n 个用户的简单二级文件系统，每次用户可保存 m 个文件，用户在一次运行中只能打开一个文件，对文件必须设置保护措施。要求做到以下几点：

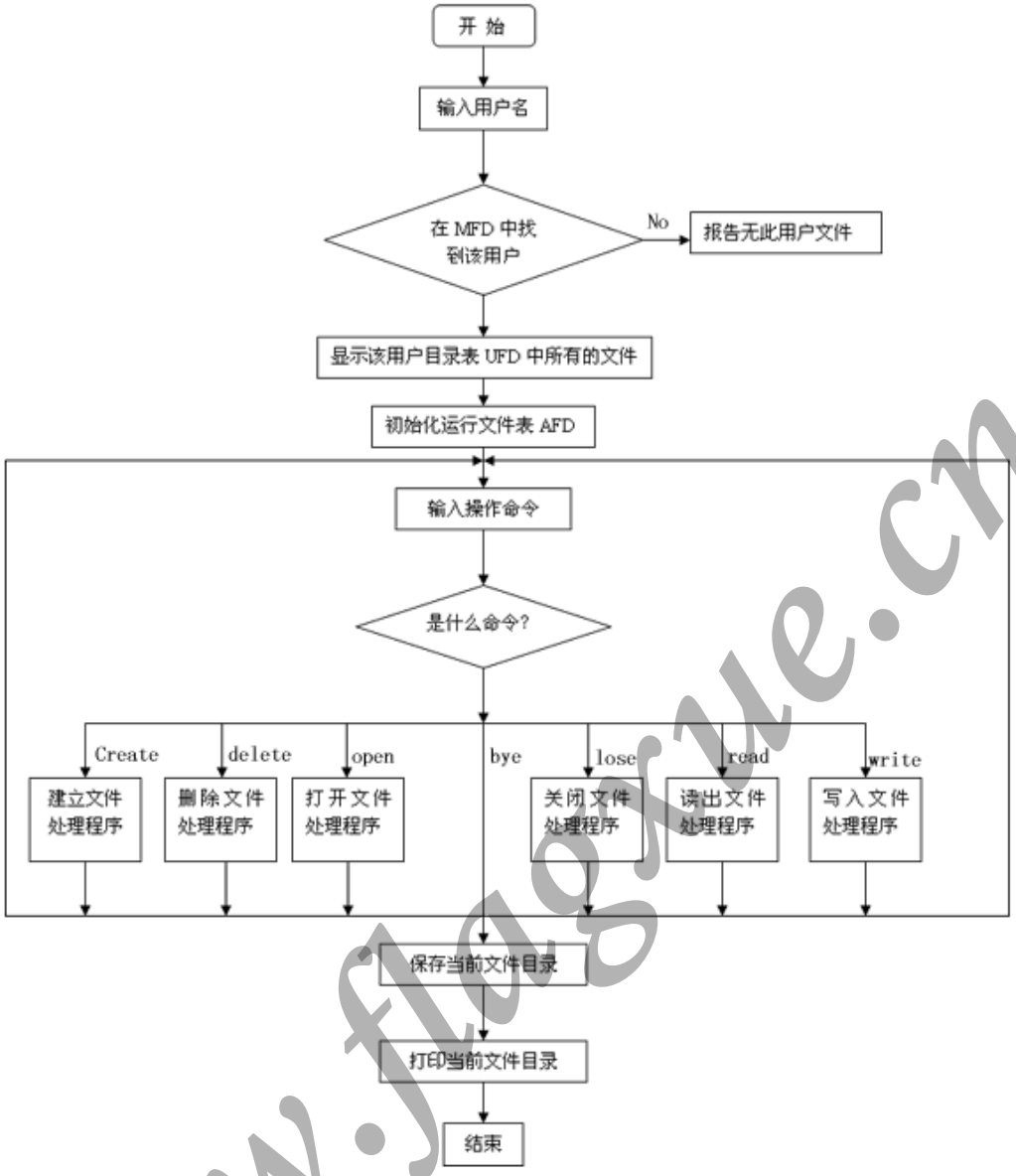
1.可以实现下列几条命令（至少 4 条）：login 用户登录；dir 列文件目录；create 创建文件；delete 删除文件；open 打开文件；close 关闭文件；read 读文件；write 写文件。

2.列目录时要列出文件名、物理地址、保护码和文件长度。

3.源文件可以进行读写保护。

三、程序设计分析

文件算法流程图



四、数据结构及符号说明

```
typedef struct ANode {//打开文件目录(运行文件目录)
    char openFile[20];    //打开文件名
    char openCode[3]; //打开保护码
    int point;   //读写指针
    struct ANode *next;

}AFD, *pAFD;
```

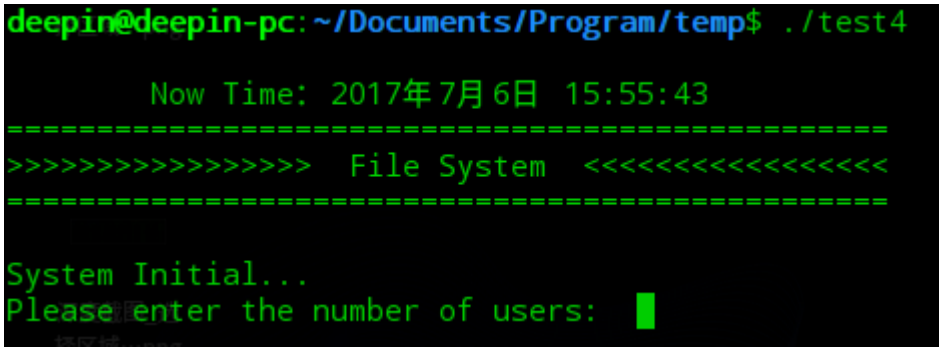
```
typedef struct UNode {//用户文件目录
    char fileName[20];
    char proCode[3]; //三位保护码，对应读写执行
    int length;
    struct UNode *next;
}UFD, *pUFD;
```

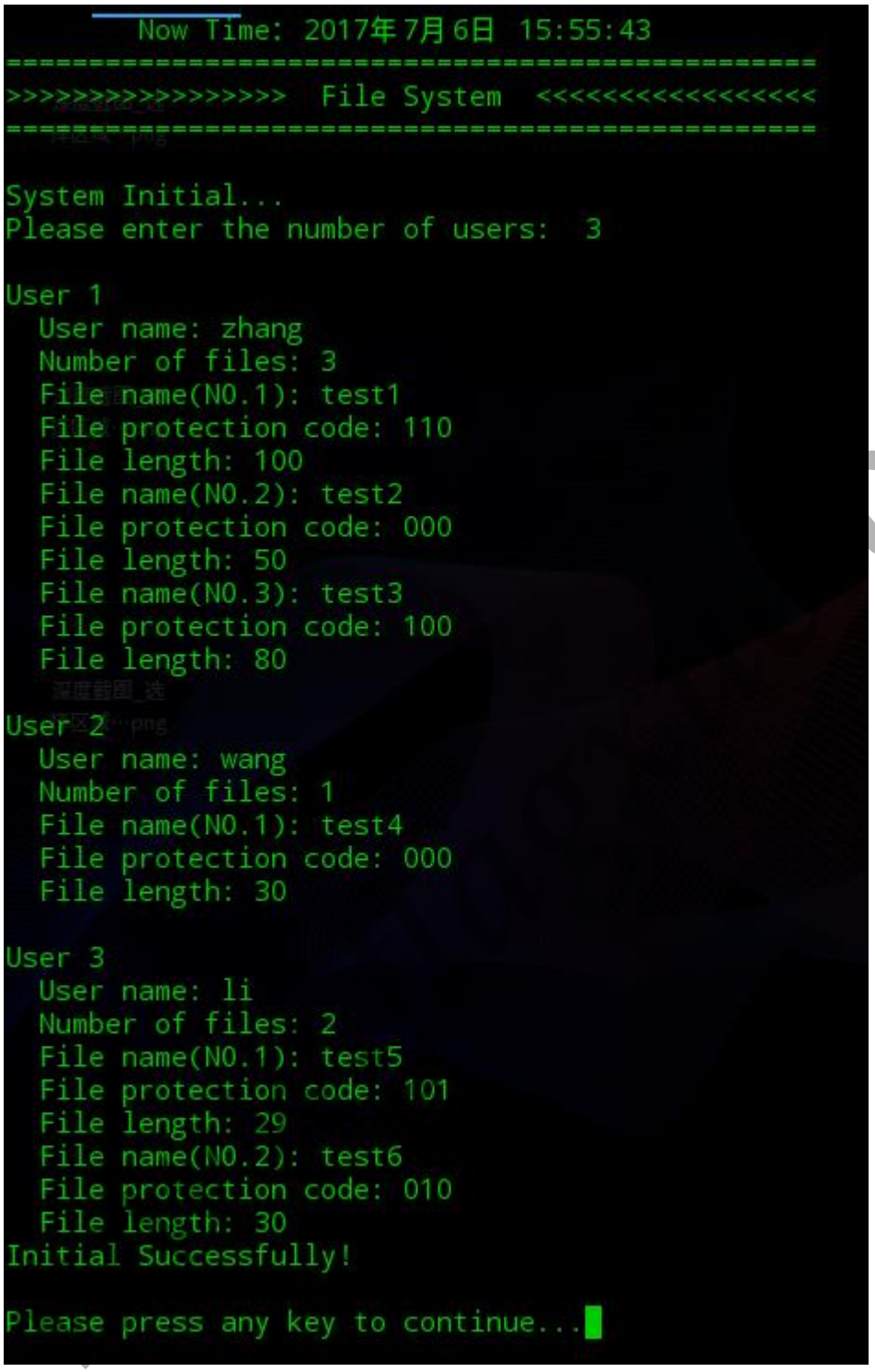
```
typedef struct MNode {    //主文件目录定义
    char userName[20];        //用户名
    struct UNode *fileMenue;  //用户文件目录
    struct MNode *next;
}MDF, *pMDF;
```

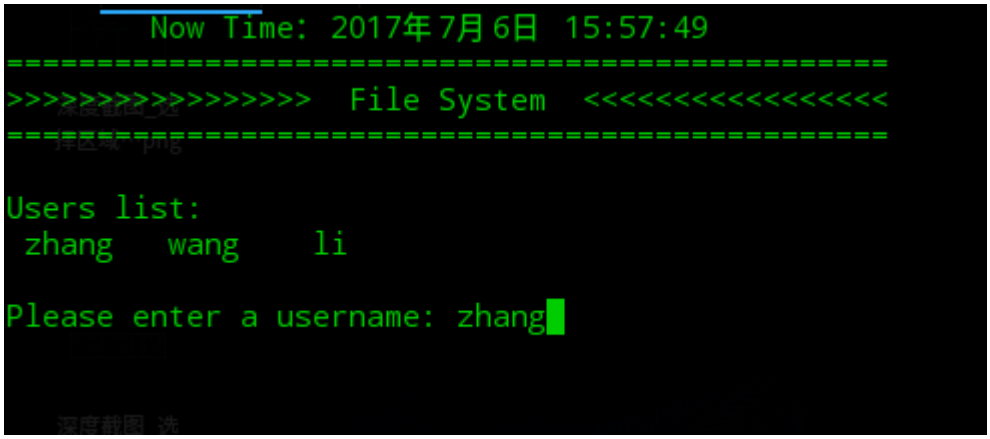

五、函数说明

	调用函数	说明
1	int main(void)	主函数
2	void initialMDF(pMDF);	初始化主文件目录
3	void initialAFD()	初始化打开文件目录
4	void displayMDF()	显示用户列表
5	void display();	显示系统信息
6	void Directory(pMDF);	显示用户文件信息
7	void Create();	创建文件
8	void Delete();	删除文件
9	void Open();	打开文件
10	void Close();	关闭文件
11	void Read();	读取文件
12	void Write();	写入文件
13	void point();	显示提示信息
14	void Relogin();	重新登陆
15	int menue();	选项菜单
16	int checkLogin(pMDF);	检查用户名
17	pUFD initialUFD();	初始化用户文件目录

六、实验数据与实验结果(见下页)

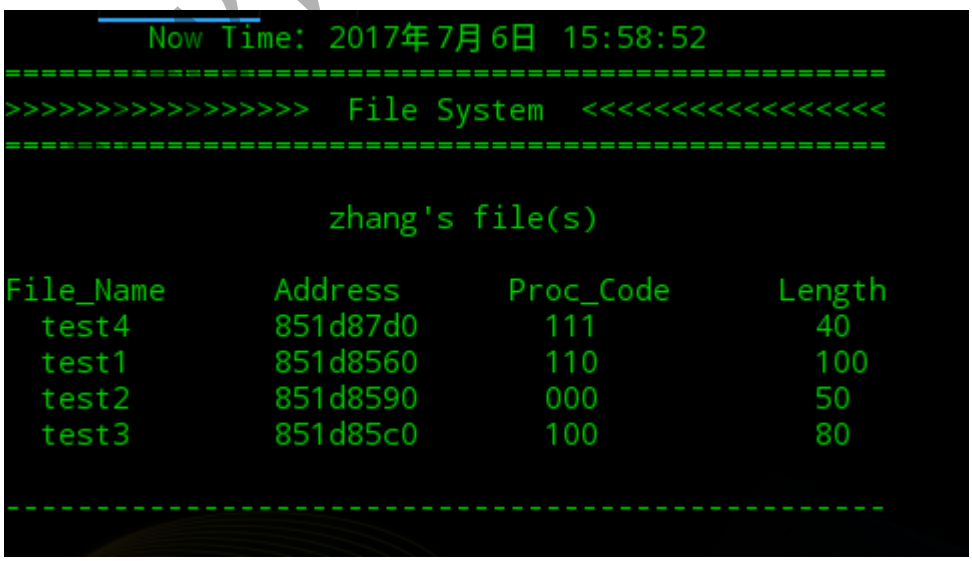


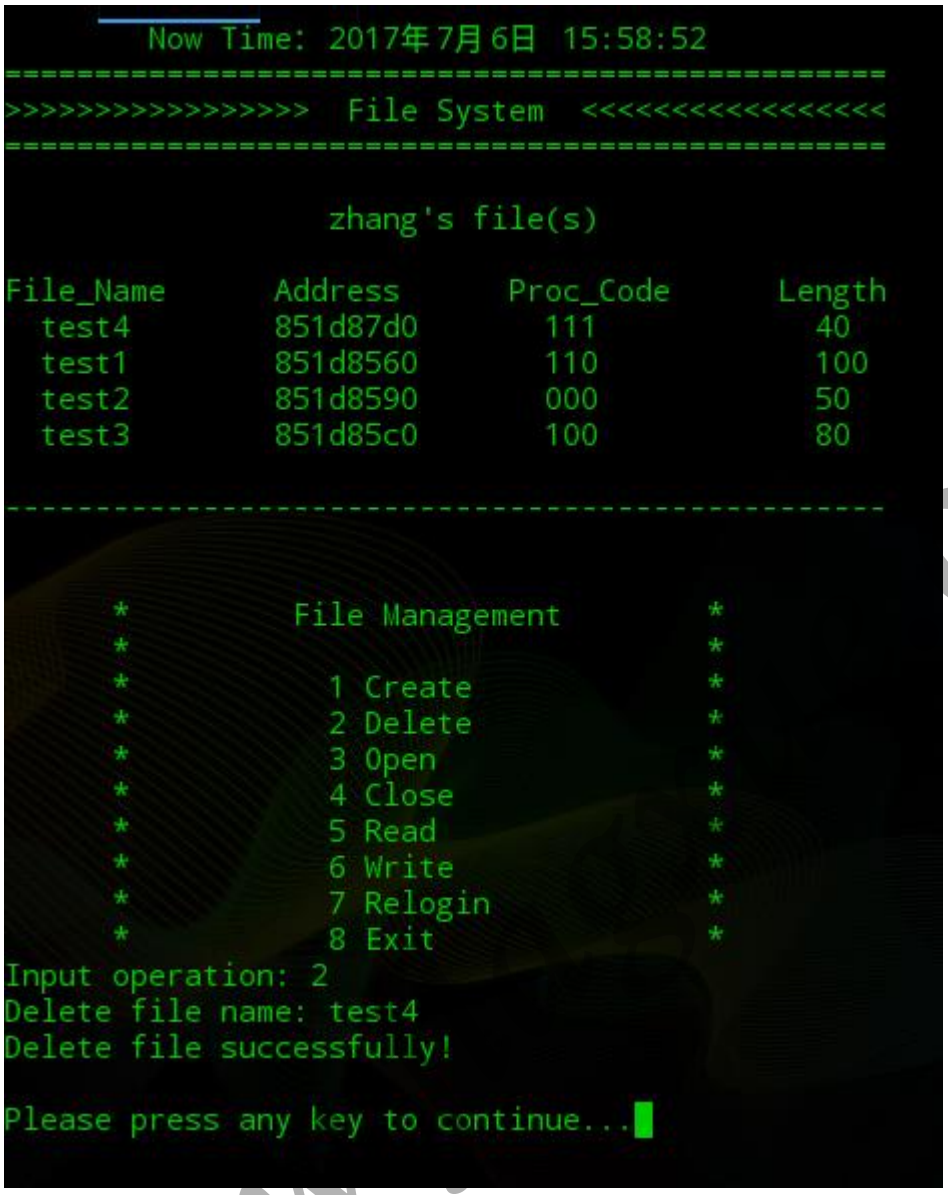

Now Time: 2017年7月6日 15:55:43
>>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<<
System Initial...
Please enter the number of users: 3
User 1
User name: zhang
Number of files: 3
File name(NO.1): test1
File protection code: 110
File length: 100
File name(NO.2): test2
File protection code: 000
File length: 50
File name(NO.3): test3
File protection code: 100
File length: 80
User 2
User name: wang
Number of files: 1
File name(NO.1): test4
File protection code: 000
File length: 30
User 3
User name: li
Number of files: 2
File name(NO.1): test5
File protection code: 101
File length: 29
File name(NO.2): test6
File protection code: 010
File length: 30
Initial Successfully!
Please press any key to continue...

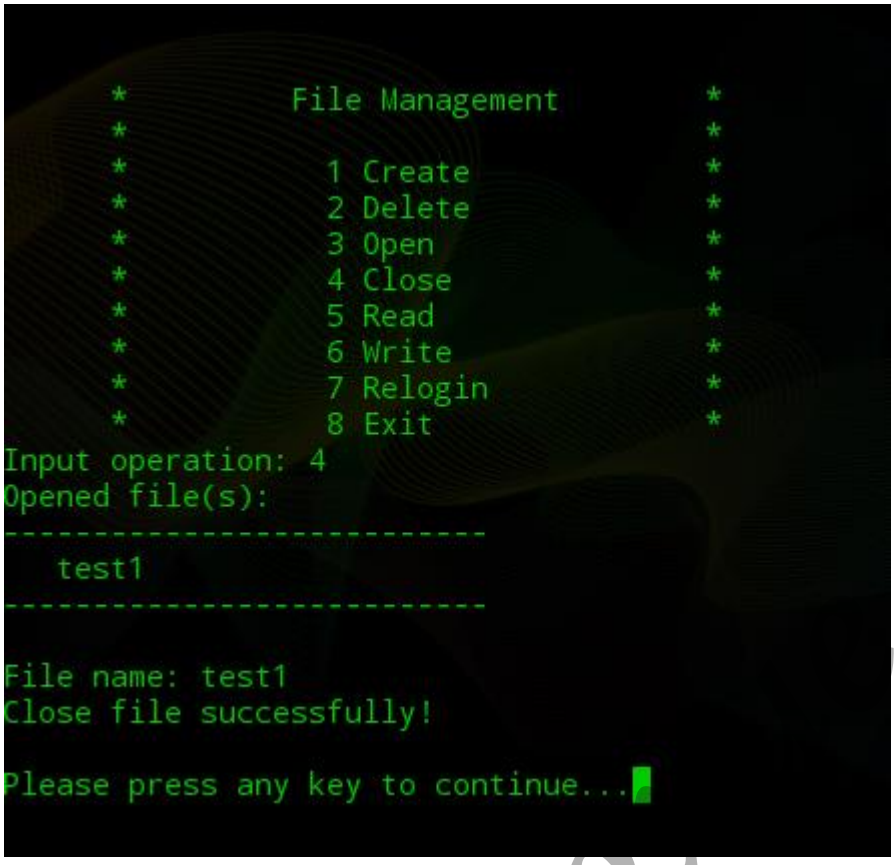

Now Time: 2017年7月6日 15:57:49
>>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<<
Users list:
zhang wang li
Please enter a username: zhang

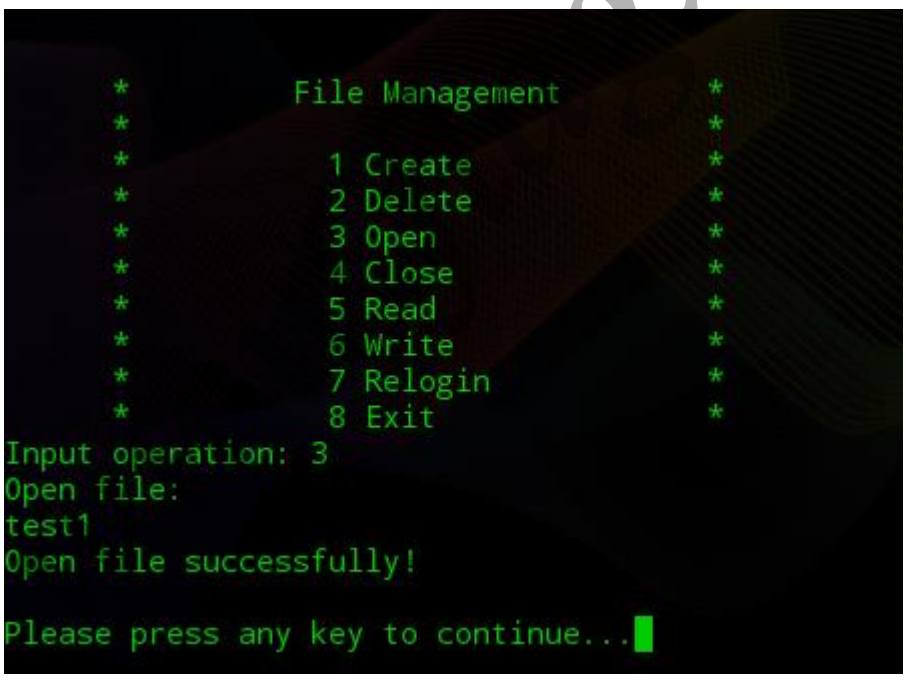

Now Time: 2017年7月6日 15:58:4
>>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<<
zhang's file(s)
File_Name Address Proc_Code Length
test1 851d8560 110 100
test2 851d8590 000 50
test3 851d85c0 100 80
File Management
1 Create
2 Delete
3 Open
4 Close
5 Read
6 Write
7 Relogin
8 Exit
Input operation:

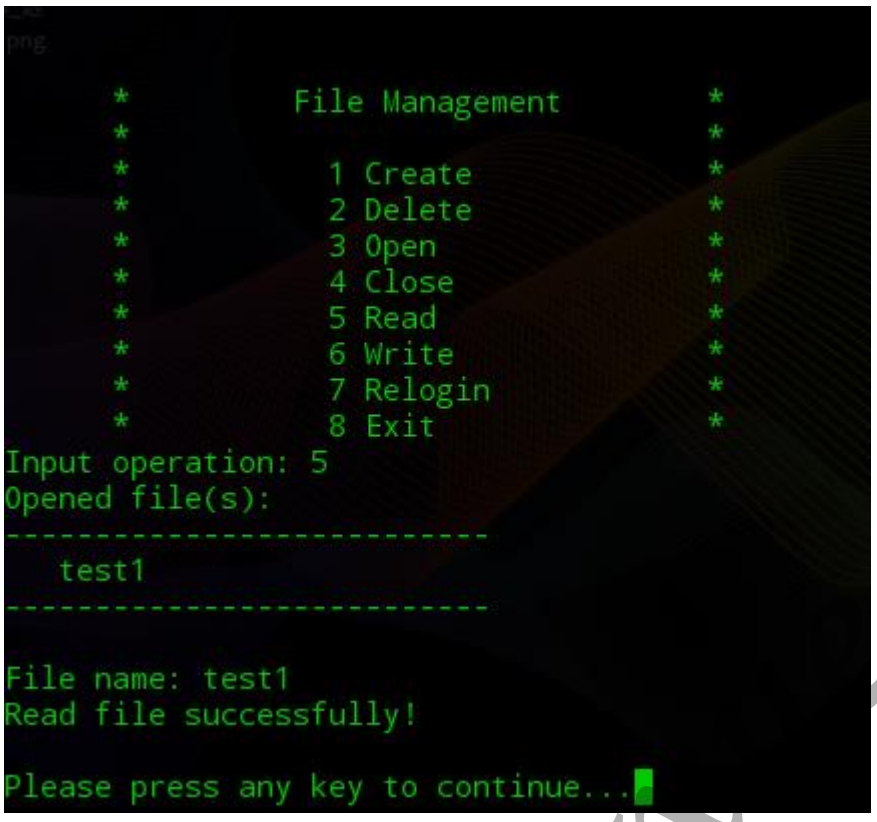

Now Time: 2017年7月6日 15:58:4
>>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<<
zhang's file(s)
File_Name Address Proc_Code Length
test1 851d8560 110 100
test2 851d8590 000 50
test3 851d85c0 100 80
* File Management *
* *
* 1 Create *
* 2 Delete *
* 3 Open *
* 4 Close *
* 5 Read *
* 6 Write *
* 7 Relogin *
* 8 Exit *
Input operation: 1
File name: test4
File protection code: 111
File length: 40
Create file test4 successfully!
Please press any key to continue...

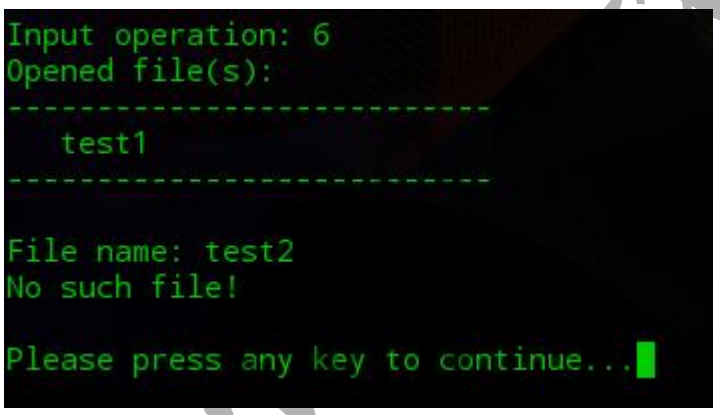

Now Time: 2017年7月6日 15:58:52
>>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<<
zhang's file(s)
File_Name Address Proc_Code Length
test4 851d87d0 111 40
test1 851d8560 110 100
test2 851d8590 000 50
test3 851d85c0 100 80


Now Time: 2017年7月6日 15:58:52
>>>>>>>>>>>>>>>> File System <<<<<<<<<<<<<<<<
zhang's file(s)
File_Name Address Proc_Code Length
test4 851d87d0 111 40
test1 851d8560 110 100
test2 851d8590 000 50
test3 851d85c0 100 80
File Management
1 Create
2 Delete
3 Open
4 Close
5 Read
6 Write
7 Relogin
8 Exit
Input operation: 2
Delete file name: test4
Delete file successfully!
Please press any key to continue...


* File Management *
* 1 Create *
* 2 Delete *
* 3 Open *
* 4 Close *
* 5 Read *
* 6 Write *
* 7 Relogin *
* 8 Exit *
Input operation: 4
Opened file(s):
test1
File name: test1
Close file successfully!
Please press any key to continue...


* File Management *
* 1 Create *
* 2 Delete *
* 3 Open *
* 4 Close *
* 5 Read *
* 6 Write *
* 7 Relogin *
* 8 Exit *
Input operation: 3
Open file:
test1
Open file successfully!
Please press any key to continue...





七、实验总结

通过编写并运行这个大工程，对文件管理和目录管理有了更深刻的了解，并且重温了 C 语言编程的一些函数。本实验很接近于实际应用，对提高自身能力很有帮助。

八、实验用程序清单

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

typedef struct ANode {//打开文件目录(运行文件目录)
    char openFile[20];    //打开文件名
    char openCode[3]; //打开保护码
    int point;   //读写指针
    struct ANode *next;

}AFD, *pAFD;

typedef struct UNode {//用户文件目录
    char fileName[20];
    char proCode[3]; //三位保护码，对应读写执行
    int length;
    struct UNode *next;
}UFD, *pUFD;

typedef struct MNode {    //主文件目录定义
    char userName[20];        //用户名
    struct UNode *fileMenue;  //用户文件目录
    struct MNode *next;
}MDF, *pMDF;

int flag;
pMDF pHead;
pMDF nowUser;
pAFD tempUser;


void display();
void initialMDF(pMDF);
void initialAFD();
void displayMDF();    //显示用户列表
void Directory(pMDF);
void Create();
void Delete();
void Open();
```

```
42  void Close();
43  void Read();
44  void Write();
45  void point();
46  void Relogin();
47  int menue();
48  int checkLogin(pMDF);
49
50  pUFD initialUFD();
51
52  void point() {
53      printf("Please press any key to continue...");
54      setbuf(stdin, NULL);
55      getchar();
56      system("clear");
57  }
58  void display() {
59      setbuf(stdin, NULL);
60      system("clear");
61      struct tm *p;
62      time_t t = time(0);
63      p = localtime(&t);
64      printf("        Now Time: %d年%d月%d日 %d:%d:%d\n", 1900 +
    p->tm_year, 1 + p->tm_mon, p->tm_mday, p->tm_hour, p->tm_min, p->tm_sec);
65      printf("=============================================\n");
66      printf(">>>>>>>>>>>>>>>>  File System  <<<<<<<<<<<<<<<<\n");
67      printf("=============================================\n");
68      printf("\n");
69  }
70
71
72
73  void initialMDF(pMDF L) {
74
75
76      display();
77      int i, n;     //n个用户
78      pMDF p;
79      L->fileMenue = NULL;
80      L->next = NULL;   //创建带头结点的空链表
81      p = L;
82
83      printf("System Initial...\n");
84      printf("Please enter the number of users:  ");
```

```
85      scanf("%d", &n);
86      for (i = 0; i < n; i++) {
87          setbuf(stdin, NULL);
88          p = p->next = (pMDF)malloc(sizeof(MDF));
89          printf("\nUser %d\n", i + 1);
90          printf("  User name: ");
91          scanf("%s", p->userName);
92          p->fileMenue = initialUFD();
93          p->next = NULL;
94      }
95
96      printf("Initial Successfully!\n\n");
97      point();
98  }
99
100 pUFD initialUFD() {
101     int i, m;    //用户文件数目
102     pUFD p, r, L;
103     L = (pUFD)malloc(sizeof(UFD));
104     L->next = NULL;
105     r = L;
106     printf("  Number of files: ");
107     scanf("%d", &m);
108     for (i = 0; i < m; i++) {
109         setbuf(stdin, NULL);
110         p = (pUFD)malloc(sizeof(UFD));
111         printf("  File name(NO.%d): ", i + 1);
112         scanf("%s", p->fileName);
113         printf("  File protection code: ");
114         scanf("%s", p->proCode);
115         printf("  File length: ");
116         scanf("%d", &p->length);
117         p->next = NULL;
118         r->next = p;
119         r = p;
120     }
121     return L->next;
122 }
123
124 void initialAFD() {
125     pAFD p, r;
126     pUFD s = nowUser->fileMenue;
127     tempUser->next = NULL;
128     r = tempUser;
```

```
129     while (s) {
130         p = (pAFD)malloc(sizeof(AFD));
131         strcpy(p->openFile, s->fileName);
132         strcpy(p->openCode, s->proCode);
133         p->point = 0;
134         p->next = NULL;
135         r->next = p;
136         r = p;
137         s = s->next;
138     }
139 
140 }
141 
142 int checkLogin(pMDF L) {
143     display();
144     char userName[20];
145     pMDF p = L->next;
146 
147     displayMDF();
148     printf("Please enter a username: ");
149     scanf("%s", userName);
150 
151     while (p) {
152 
153         if (strcmp(userName, p->userName) == 0) {
154             printf("Login in successfully!\n");
155             sleep(1);
156             nowUser = p;
157             return 1;
158         }
159         p = p->next;
160     }
161     printf("Invalid user!\n");
162     sleep(1);
163     return 0;
164 }
165 
166 void displayMDF() {
167     printf("Users list: \n");
168     pMDF p = pHead->next;
169     while (p) {
170         printf(" %s\t", p->userName);
171         p = p->next;
172     }
```

```
173     printf("\n\n");
174 }
175
176 void Directory(pMDF L) {
177     display();
178     pUFD p = nowUser->fileMenue;
179     printf("                  %s's file(s)                    \n\n",
    L->userName);
180     printf("File_Name      Address      Proc_Code      Length\n");
181     while (p) {
182         printf("  %s\t       ", p->fileName);
183         printf("%x        ", p);
184         printf("%s\t      ", p->proCode);
185         printf("%d\n", p->length);
186         p = p->next;
187     }
188     printf("\n");
189     printf("-----------------------------------------\n\n");
190 }
191 void Create() {
192     setbuf(stdin, NULL);
193     char fileName[20];
194
195     pUFD s = nowUser->fileMenue;
196     pUFD p = (pUFD)malloc(sizeof(UFD));
197     pAFD pp = (pAFD)malloc(sizeof(AFD));
198     pUFD temp = nowUser->fileMenue;
199     pAFD tempp = tempUser->next;
200
201     printf("File name: ");
202     scanf("%s", fileName);
203
204     while (s) {
205
206         if (strcmp(fileName, s->fileName) == 0) {
207             printf("Error: The file name already exists!");
208             return;
209         }
210
211         s = s->next;
212     }
213
214     strcpy(p->fileName, fileName);
215     strcpy(pp->openFile, fileName);
```

```
216         setbuf(stdin, NULL);
217         printf("  File protection code: ");
218         scanf("%s", p->proCode);
219         strcpy(pp->openCode, p->proCode);
220         printf("  File length: ");
221         scanf("%d", &p->length);
222         pp->point = 0;
223
224         nowUser->fileMenue = p;
225         p->next = temp;
226         tempUser->next = pp;
227         pp->next = tempp;
228
229         printf("Create file %s successfully!\n\n", fileName);
230         point();
231     }
232     void Delete() {
233         char fileName[20];
234         pUFD p = nowUser->fileMenue;
235         pAFD pp = tempUser->next;
236         pUFD temp;
237         pAFD tempp;
238         printf("Delete file name: ");
239         scanf("%s", fileName);
240
241         if (strcmp(fileName, p->fileName) == 0) {
242             nowUser->fileMenue = nowUser->fileMenue->next;
243             tempUser->next = pp->next;
244             free(p);
245             free(pp);
246             printf("Delete file successfully!\n\n");
247             point();
248             return;
249         }
250
251         while (p->next) {
252             if (strcmp(fileName, p->next->fileName) == 0) {
253                 temp = p->next;
254                 tempp = pp->next;
255                 p->next = p->next->next;
256                 pp->next = pp->next->next;
257                 free(temp);
258                 free(tempp);
259                 printf("Delete file successfully!\n\n");
```

```
260             point();
261             return;
262         }
263         p = p->next;
264         pp = pp->next;
265     }
266     printf("Error! No such file!\n\n");
267     printf("Please press any key to continue...");
268     setbuf(stdin, NULL);
269     getchar();
270     system("clear");
271     return;
272 }
273 void Open() {
274     char fileName[20];
275     pAFD p = tempUser->next;
276 
277     printf("Open file: \n");
278     scanf("%s", fileName);
279 
280     while (p) {
281         if (strcmp(fileName, p->openFile) == 0) {
282             if (!p->point) {
283                 printf("Open file successfully!\n\n");
284                 p->point = 1;
285             }
286             else
287                 printf("The file is already opened!\n\n");
288             point();
289             return;
290         }
291         p = p->next;
292     }
293     printf("No such file!\n\n");
294     point();
295 }
296 
297 void Close() {
298     char fileName[20];
299     pAFD p = tempUser->next;
300     printf("Opened file(s): \n");
301     printf("------------------------\n");
302     while (p) {
303         if (p->point)
```

```
304                 printf("   %s", p->openFile);
305             p = p->next;
306         }
307         printf("\n-------------------------\n\n");
308         p = tempUser->next;
309         printf("File name: ");
310         scanf("%s", fileName);
311         while (p) {
312             if (strcmp(fileName, p->openFile) == 0) {
313                 if (p->point) {
314                     printf("Close file successfully!\n\n");
315                     p->point = 0;
316                 }
317                 else
318                     printf("The file has not been opened!\n\n");
319                 point();
320                 return;
321             }
322             p = p->next;
323         }
324         printf("No such file!\n\n");
325         point();
326 }
327 void Read() {
328         char fileName[20];
329         pAFD p = tempUser->next;
330         printf("Opened file(s): \n");
331         printf("-------------------------\n");
332         while (p) {
333             if (p->point)
334                 printf("   %s", p->openFile);
335             p = p->next;
336         }
337         printf("\n-------------------------\n\n");
338         p = tempUser->next;
339         printf("File name: ");
340         scanf("%s", fileName);
341
342         while (p) {
343             if (strcmp(fileName, p->openFile) == 0) {
344                 if (p->point) {
345                     if (p->openCode[0] == '1')
346                         printf("Read file successfully!\n\n");
347                     else
```

```
348                         printf("The file is procted!\n\n");
349                 }
350                 else
351                     printf("The file has not been opened!\n\n");
352                 point();
353                 return;
354 
355             }
356             p = p->next;
357         }
358         printf("No such file!\n\n");
359         point();
360 }
361 void Write() {
362         char fileName[20];
363         pAFD p = tempUser->next;
364         printf("Opened file(s): \n");
365         printf("-------------------------\n");
366         while (p) {
367             if (p->point)
368                 printf("   %s", p->openFile);
369             p = p->next;
370         }
371         printf("\n-------------------------\n\n");
372         p = tempUser->next;
373         printf("File name: ");
374         scanf("%s", fileName);
375 
376         while (p) {
377             if (strcmp(fileName, p->openFile) == 0) {
378                 if (p->point) {
379                     if (p->openCode[1] == '1')
380                         printf("Write file successfully!\n\n");
381                     else
382                         printf("The file is protected!\n\n");
383                 }
384                 else
385                     printf("The file has not been opened!\n\n");
386                 point();
387                 return;
388             }
389             p = p->next;
390         }
391         printf("No such file!\n\n");
```

```
392     point();
393 }
394
395 void Relogin() {
396     flag = 1;
397     pAFD delete, next;
398     delete = tempUser->next;
399     while (delete) {
400         next = delete->next;
401         free(delete);
402         delete = next;
403     }
404
405 }
406 int menue() {
407
408     setbuf(stdin, NULL);
409     int choose;
410     do {
411         printf("\n");
412         printf("      *         File Management         *\n");
413         printf("      *                                 *\n");
414         printf("      *           1 Create              *\n");
415         printf("      *           2 Delete              *\n");
416         printf("      *           3 Open                *\n");
417         printf("      *           4 Close               *\n");
418         printf("      *           5 Read                *\n");
419         printf("      *           6 Write               *\n");
420         printf("      *           7 Relogin             *\n");
421         printf("      *           8 Exit                *\n");
422
423         fflush(stdin);
424         printf("Input operation: ");
425         scanf("%d", &choose);
426         if (choose != 1 && choose != 2 && choose != 3 && choose != 4 &&
    choose != 5 && choose != 6 && choose != 7 && choose != 8 && choose != 9)
    {
427             printf("Invalid input! Please input again!\n");
428             sleep(3);
429         }
430     } while (choose != 1 && choose != 2 && choose != 3 && choose != 4 &&
    choose != 5 && choose != 6 && choose != 7 && choose != 8 && choose != 9);
431     return choose;
432 }
```

```
433
434 int main(void) {
435     int choose;
436     pHead = (pMDF)malloc(sizeof(MDF));
437     tempUser = (pAFD)malloc(sizeof(AFD));
438     initialMDF(pHead);
439
440     while (1) {
441         flag = 0;
442         if (checkLogin(pHead)) {
443             initialAFD();
444             while (1) {
445                 setbuf(stdin, NULL);
446                 Directory(nowUser);
447                 choose = menue();
448                 switch (choose) {
449                 case 1:
450                     Create();
451                     break;
452                 case 2:
453                     Delete();
454                     break;
455                 case 3:
456                     Open();
457                     break;
458                 case 4:
459                     Close();
460                     break;
461                 case 5:
462                     Read();
463                     break;
464                 case 6:
465                     Write();
466                     break;
467                 case 7:
468                     Relogin();
469                     break;
470                 case 8:
471                     exit(1);
472                     break;
473                 }
474                 if (flag == 1)
475                     break;
476             }
```

```
477
478             }
479         }
480         return 0;
481                                         }
```

评价表格

考核标准	得分
（1）正确理解和掌握实验所涉及的概念和原理（20%）；	
（2）按实验要求合理设计数据结构和程序结构（20%）；	
（3）运行结果正确（20%）；	
（4）认真记录实验数据，原理及实验结果分析准确（20%）；	
(5)实验过程中,具有严谨的学习态度和认真、踏实、一丝不苟的科学作风(10%);	
（7）实验报告规范（10%）。	
合计	