操 作 系 统 原 理

实 验 报 告

实验题目： 实验 5：LRU 页面置换算法

实验时间： 2022 年 12 月 21 日

学号姓名： 吕强 22129390

实验目的和要求

1.加深对面置换的概念和算法的理解。

2.深入了解 LRU 页面置换算法。

实验内容

在 Visual C++ 6.0 集成开发环境下使用 C 或 C++语言，设计并实现一个 LRU 页面置换算法的应用程序。

实验原理与提示

（1）实现 LRU 置换算法

（2）页面序列从指定的文本文件（TXT 文件）中取出

（3）输出：第一行：每次淘汰的页面号，第二行：显示缺页的总次数

实验程序

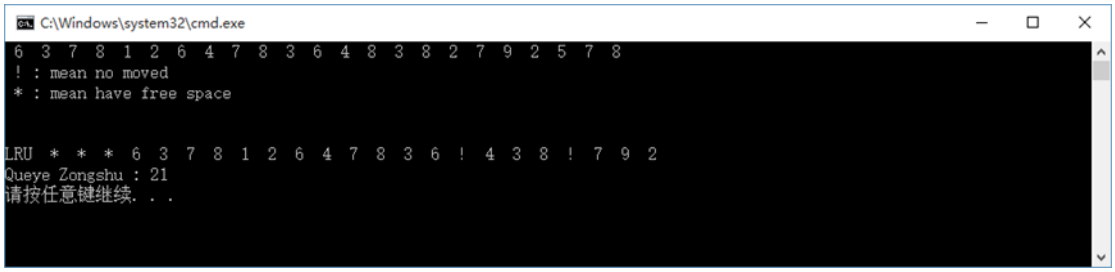
```
#include "stdio.h"
#include "stdlib.h" //产生随机数所需头文件
#include "malloc.h"
#include "iostream"
#define null 0
#define len sizeof(struct page)    //作业页表结构体

struct page
{
	int num;//记录页面号
	int tag;
	struct page *next;
};

struct page *create(int n)
```

```
{
	int count = 1;
	struct page *p1, *p2, *head;
	head = p2 = p1 = (struct page *)malloc(len);
	p1->tag = -1; p1->num = -1;
	while (count<n)
	{
		count++;
		p1 = (struct page *)malloc(len);
		p1->tag = -1; p1->num = -1;
		p2->next = p1;
		p2 = p1;
	}
	p2->next = null;
	return(head);
}

void LRU(int* array,int n)
{
	int count = 0, *p = array;
	struct page *head, *cp, *dp, *rp, *New, *endp;
	head = create(n);
	while (*p != -1)
	{
		cp = dp = rp = endp = head;
		for (; endp->next != null;) endp = endp->next;
		for (; cp->num != *p&&cp->next != null;)
		{
			rp = cp; cp = cp->next;
		}
		if (cp->num == *p)
		{
			printf(" ! ");
			if (cp->next != null)
			{
				if (cp != head)
					rp->next = cp->next;
				else head = head->next;
			}
			endp->next = cp;
			cp->next = null;
		}
		else
		{
```

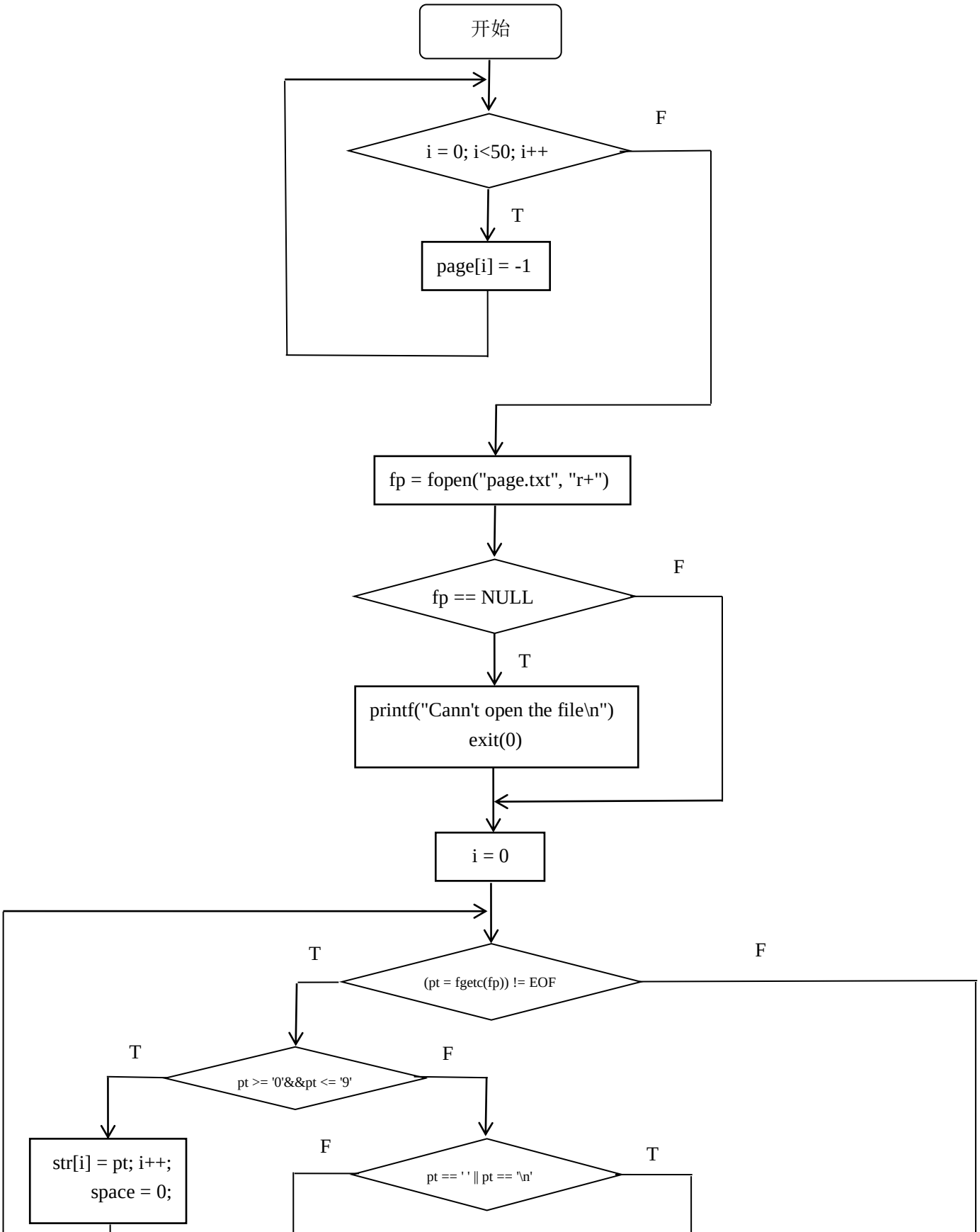
```
                count++;
                cp = rp = head;
                for (; cp->tag != -1 && cp->next != null;) cp = cp->next;
                if (cp->tag == -1)
                {
                        printf(" * ");
                        cp->num = *p;
                        cp->tag = 0;
                }
                else
                {
                        New = (struct page *)malloc(len);
                        New->num = *p;
                        New->tag = 0;
                        New->next = null;
                        cp->next = New;
                        dp = head;
                        head = head->next;
                        printf(" %d ", dp->num);
                        free(dp);
                }
            }
            p++;
        }
        printf("\nQueye Zongshu : %d \n", count);
}

void main()
{
        FILE *fp;
        char pt;
        char str[10];
        int i, j = 0;
        int page[50], space = 0;
        for (i = 0; i<50; i++)
                page[i] = -1;
        fp = fopen("page.txt", "r+");
        if (fp == NULL)
        {
                printf("Cann't open the file\n");
                exit(0);
        }
        i = 0;
        while ((pt = fgetc(fp)) != EOF)
```

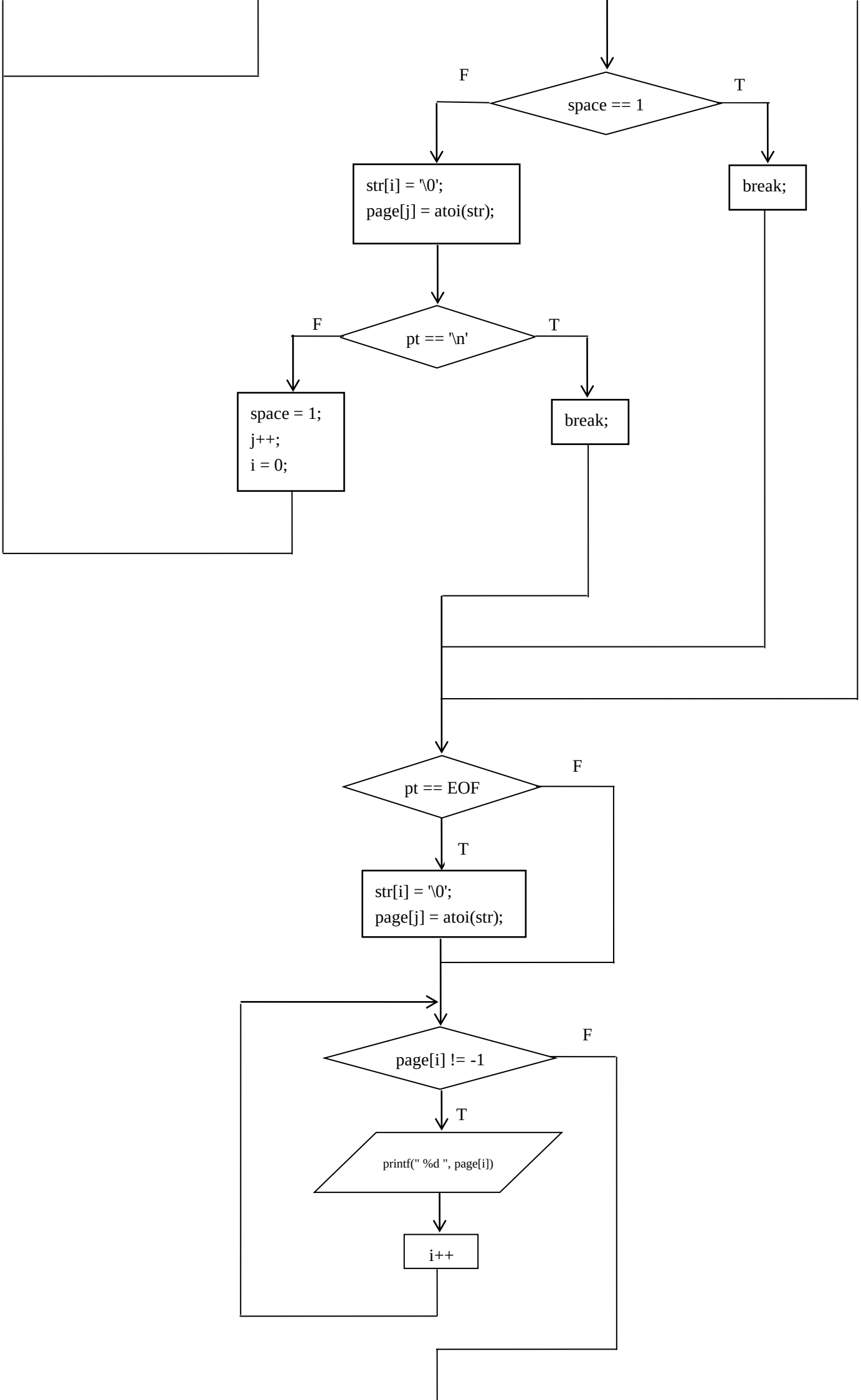
```
    {
        if (pt >= '0'&&pt <= '9')
        {
            str[i] = pt; i++;
            space = 0;
        }
        else
        {
            if (pt == ' ' || pt == '\n')
            {
                if (space == 1) break;
                else
                {
                    str[i] = '\0';
                    page[j] = atoi(str);
                    if (pt == '\n') break;
                    else
                    {
                        space = 1;
                        j++;
                        i = 0;
                    }
                }
            }
        }
    }
    if (pt == EOF) { str[i] = '\0'; page[j] = atoi(str); }
    i = 0;
    while (page[i] != -1) { printf(" %d ", page[i]); i++; }
    fclose(fp);
    printf("\n");
    printf(" ! : mean no moved \n * : mean have free space \n\n");//每次淘汰的页面号
    printf("\nLRU ");//缺页的总次数
    LRU(page, 3);
}
```

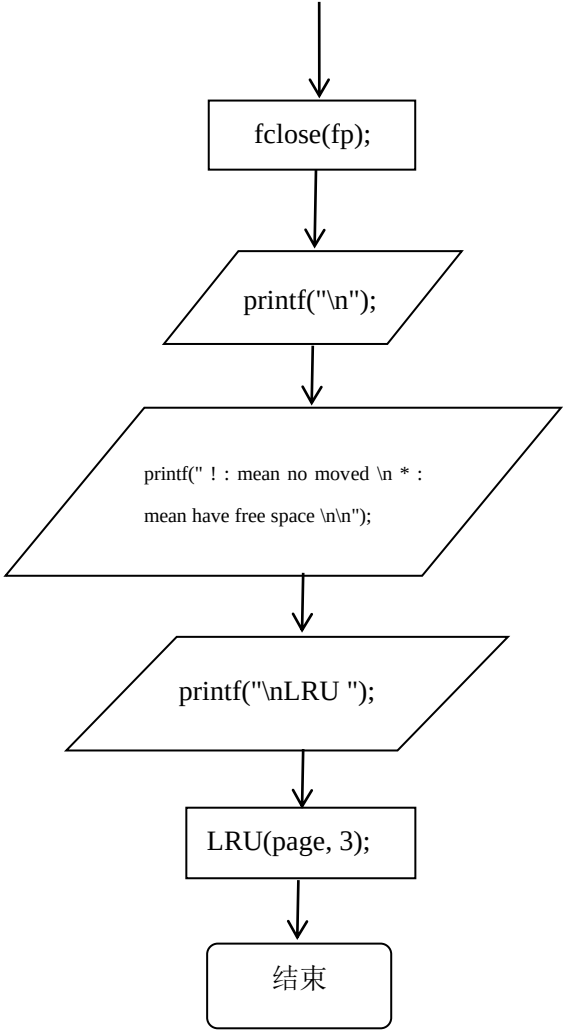
运行结果截图


C:\Windows\system32\cmd.exe
6 3 7 8 1 2 6 4 7 8 3 6 4 8 3 8 2 7 9 2 5 7 8
! : mean no moved
* : mean have free space
LRU * * * 6 3 7 8 1 2 6 4 7 8 3 6 ! 4 3 8 ! 7 9 2
Queye Zongshu : 21
请按任意键继续. . .

主函数流程图


开始
i = 0; i<50; i++
F
T
page[i] = -1
fp = fopen("page.txt", "r+")
fp == NULL
F
T
printf("Cann't open the file\n")
exit(0)
i = 0
T
F
(pt = fgetc(fp)) != EOF
T
F
pt >= '0'&&pt <= '9'
str[i] = pt; i++;
space = 0;
F
T
pt == ' ' || pt == '\n'


F
space == 1
T
str[i] = '\0';
page[j] = atoi(str);
break;
F
pt == '\n'
T
space = 1;
j++;
i = 0;
break;
pt == EOF
F
T
str[i] = '\0';
page[j] = atoi(str);
page[i] != -1
F
T
printf(" %d ", page[i])
i++



分析讨论

通过本次实验,了解到 LRU 算法(最近最久未使用更新算法)的基本原理,LRU 算法与每个页面最后被访问的时间有关,该算法赋予每个页面一个访问字段,用来记录一个页面自上次被访问以来所经历的时间 t,当必须淘汰一个页面时,此算法将内存中 t 值最大的页面给予淘汰.

教师评语及成绩