电子科技大学

计算机专业类课程

实验报告

课程名称：操作系统

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：冯铭扬

学　　号：2013060109023

指导教师：薛瑞尼

日　　期：2016　年　6　月　10　日

电 子 科 技 大 学

实　　验　　报　　告

实验四

一、实验名称：混合索引逻辑地址到物理地址映射

二、实验学时：4

三、实验内容和目的：

1 条件：自定义混合索引 inode 结构

- 必须包括一次，二次，和三次间接块
- 逻辑块 n 对应物理块 n

2 输入：文件逻辑地址

3 输出

- 输出 inode 详细信息（间接块不展开）

物理地址（物理块号，块内偏移）

四、实验器材（设备、元器件） windows ＋ vc＋＋

五、实验数据及结果分析：

实验代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```c
#include <unistd.h>
#define SIZE 256

int main(int argc, char *argv[])
{

    // VARIABLE DECLARATIONS
    int addressFile,backingStore;            // file descriptors
    char *file= argv[1];

    char ch,ct, input[1000], output;
    int logicalAddress, physicalAddress;
    int i=0, j=0;


    // LOGICAL MEMORY
    int p;                                // page-number: used as an index
into a page table
    int d;                                   // page-offset

    // PHYSICAL MEMORY
    int f;                                   // frame-number: base address of
each page in physical memory
    char frames[SIZE*SIZE];                  //  physical memory where frame
0 is from frames[0] to frames[255]
    int frametable[SIZE];                    // one free (-1) or allocated (0)
flag entry for every frame 0 to 256
    int start, current;                      // var to handle page faults
    int offset, pagefault=0;
    int freeFrame=-1;


    // PAGE TABLE
    int pagetable[SIZE];                     // page table where table[p] = f
    for (j=0;j<SIZE;j++)                     // flushing page table,
frametable
    {
        pagetable[j] = -1;
        frametable[j] = -1;
    }

    // READING LOGICAL ADDRESS FROM FILE

    addressFile = open(file,O_CREAT);
```

```
    backingStore = open("BACKING_STORE.txt",O_RDONLY);

    if(addressFile != -1)
    {
        while(read(addressFile, &ch, sizeof(char)) != 0)   //  read()
returns the number of bytes read and 0 for end of file (EOF)
        {
            if(ch != '\n')
            {
                input[i] = ch;
                i++;
            }
            else
            {
                logicalAddress =atoi(input);
                //          EXTRACT PAGE NUMBER AND OFFSET
                p = (logicalAddress & 0x0000ff00UL) >>  8;
                d = (logicalAddress & 0x000000ffUL);

                printf("\nlogicalAddress: %d, p: %d, d: %d",
logicalAddress,p,d);

                //          ADDRESS TRANSLATION THROUGH PAGE TABLE

                //          pagetable-hit, obtain frame number
                if(pagetable[p]!= -1){

                    f = pagetable[p];
                    physicalAddress = (f * SIZE) + d;
                    printf("\nphysicalAddress: %d, f: %d",
physicalAddress,f);

                }
                //          pagetable-miss, page-fault
                else
                {
                    pagefault++;
                    //              locate free frame (-1) in physical memory
                    for (j=0;j<SIZE;j++)
                    {
                        if(frametable[j]==-1)
                        {
                            freeFrame = j;
                            break;
                        }
                    }
```

```
                //              read page from backing-store into the
available frame in the physical memory
                if(backingStore != -1)
                {

                    offset=0;
                    start = SIZE * p;
                    current=lseek(backingStore, start, SEEK_SET);
                    while((offset < SIZE)&&(current))
                    {
                        current = read(backingStore, &ct, sizeof(char));
                        frames[freeFrame*offset] = ct;
                        offset++;
                    }
                }
                else
                {
                    printf("Backing-Store Does not exist!");
                    close(backingStore);
                    close(addressFile);
                    return 0;
                }
                //              update pagetable, frametable
                pagetable[p] = freeFrame;
                frametable[freeFrame] = 0;

                physicalAddress = (freeFrame * SIZE) + d;
                printf("\nphysicalAddress: %d, freeFrame: %d",
physicalAddress, freeFrame);
            }

            //              READ CHAR STORED AT THE PHYSICAL ADDRESS
            output = frames[physicalAddress];
            printf("\nByte value stored at
physicalAddress %d: %c\n",physicalAddress, output);

            memset(input,0,sizeof(input));
            i=0;

        }
    }
    printf("\nTotal Page Faults: %d",pagefault);
}
else
```

```
        printf("Addresses File Does not exist!");

    close(backingStore);
    close(addressFile);
    return 0;
}
```