实验8 文件管理

1. 实验目的
2. 熟悉Linux 文件系统的文件和目录结构。
3. 掌握文件系统的基本特征。
4. 掌握常用的文件操作函数。
5. 预备知识
6. incldue<stdio.h>下的文件函数介绍
7. fopen()函数

**功能：**用于打开文件。

**函数原型：**FILE \* fopen(const char \*pathname, const char \*method);

**参数：**

pathname:被打开文件的文件路径以及文件名。

method：打开文件的方式。

具体方式如下：

“r" 或 ”rb" 以只读方式打开文件。

“w" 或 ”wb" 以写方式打开文件，新内容会覆盖原本内容。

“a” 或 “ab” 以写方式打开文件，新内容追加在文件末尾。

**返回值**：文件打开成功，指向该文件指针就会被返回。如果文件打开失败则返回NULL。

1. fread()函数

**功能：**用于读取数据。

**函数原型：**int fread ( void \*buffer, int size,int count, FILE \*stream) ;

**参数：**

buffer:用于接收数据的内存地址。

size:要读写的单字节数，单位是字节。

count：要读取多少个数据。

stream：指向要读取文本的指针。

**返回值：**返回实际读取的数据的个数，应返回值与count相同。

1. fwrite()函数

**功能：**向一个文本写入数据。

**函数原型：**int fwrite ( void \*buffer, int size,int count, FILE \*stream) ;

**参数：**

buffer： 一般要写入的数据，事先先写到缓存buffer中，当需要写入文本时，数据从此一个个写入。

size： 要写入内容的单字节数。

count： 写入数据的个数。

stream： 目标文本指针，即要存在那个文本中。

**返回值：** 返回写入数据的个数。

1. fclose()函数

**功能：** 关闭一个流。

**函数原型：** int fclose( FILE \*file )

**返回值：** 如果流成功关闭，fclose 返回 0，否则返回-1。

1. fseek()函数

**功能：** 它在文件流里为下一次读写操作指定位置。

**函数原型：** int fseek(FILE \*stream, long offset, int fromwhere)；

**参数：**

stream: 文件指针。

offset： 偏移量，就是相当于当前位置，向左（右）移动几位。正数表示右向偏移，负数表示左向偏移。

fromwhere： 定义文件中哪里开始偏移，取值可为：SEEK\_CUR（当前位置）、 SEEK\_END （文件结尾）或 SEEK\_SET（文件开头）。

其中SEEK\_SET,SEEK\_CUR和SEEK\_END依次为0，1和2。

**返回值**： 如果执行成功，函数返回0。如果执行失败，函数返回一个非0值。

1. 实验内容
2. 新建一个文件src.txt和一个文件dsrc.txt，在src.txt文件里输入：

I LOVE SHANGHAI

我爱上海

保存src.txt文件。通过编写内容复制到dsrc.txt里，并且通过变量flag来决定是否打开dsrc,txt文件进行查询。

**提示：实验需要预先创建好src.txt和dsrc.txt这两个文件，并在src.txt文件中输入内容，dsrc.txt保持为空即可**。

#include<stdio.h>

#include<stdlib.h>

//设置缓冲区块的大小值为 1，用于读取（写入）数据都是从缓冲区中获取

#define Buffer\_size 1

//定义待复制的原文件

#define src\_file\_name "src.txt"

//定义目标文件

#define dst\_file\_name "dsrc.txt"

//定义偏移量为0

#define OFFSET 0

int main()

{

FILE \*src\_file,\*dst\_file;

unsigned char src\_buff[Buffer\_size];//定义原文本的缓冲区

unsigned char dst\_buff[Buffer\_size];//定义目标文本的缓冲区

int real\_read\_len = 0;//真正读取数据的个数

char flag;//当文件copy成功时，判断是否要查看两个文件内容的标志号

src\_file=fopen(src\_file\_name,"rb");//以读方式打开原文本

dst\_file=fopen(dst\_file\_name,"wb");//以写方式打开目标文本

//如果fopen()打开失败，则会返回NULL值

if(src\_file == NULL||dst\_file == NULL)

{ printf("src\_file or dst\_file open error!\n");

}

//开始读取数据，先把数据读到src\_buff缓冲区中，之后再写入dst\_file文本中

while((real\_read\_len=fread(src\_buff,1,sizeof(src\_buff),src\_file))>0)

{ //把读到的数据写入dst\_file文本中

fwrite(src\_buff,1,real\_read\_len,dst\_file);

}

fclose(dst\_file);

printf("copy已经完成，请问是否需要查看，输入1为查看，输入0为退出。\n");

scanf("%c",&flag);

if(flag =='1')

{ printf("this is src\_file:\n");

//因为上面已经读过src\_file文本了，所以相应的文本指针已经读到文本末尾了，此时需要再次读取数据，则需要把相应的文本指针指回开头，

//又因为偏移量OFFSET为0，SEEK\_SET意思是指文件开头，所以fseek（）函数则是把指针指向文件开头，并偏移0量。

fseek(src\_file,OFFSET,SEEK\_SET);

//输出src\_file中的数据

while((real\_read\_len=fread(src\_buff,1,sizeof(src\_buff),src\_file))>0)

{

printf("%s",src\_buff);

}

printf("\nthis is dst\_file:\n");

//以读方式打开目标文本并输入里面的内容

dst\_file=fopen(dst\_file\_name,"rb");

while((real\_read\_len=fread(dst\_buff,1,sizeof(dst\_buff),dst\_file))>0)

{ printf("%s",dst\_buff);}

}else if(flag == '0') printf("退出\n");

else printf("输入有误，系统自动退出\n");

fclose(src\_file);

fclose(dst\_file);

return 0;

}

**运行结果：**

copy已经完成，请问是否需要查看，输入1为查看，输入0为退出。

1

this is src\_file:

I LOVE SHANGHAI

我爱上海

this is dst\_file:

I LOVE SHANGHAI

我爱上海

1. 在当前用户目录下创建数据文件student.txt，文件的内部信息存储格式为Sname:S#:Sdept:Sage:Ssex，即“姓名:学号:学院:年龄:性别”，每行一条记录，输入不少于10条学生记录，其中包括学生本人记录。编写程序实现从文件中查找Sdept字段值为“网络工程技术”的文本行，输出到文件csstudent.txt中，保存时各字段顺序调整为S#:Sname:Sage: Ssex:Sdept。

**提示**：从终端读入一个文本行到[字符串](https://so.csdn.net/so/search?q=%E5%AD%97%E7%AC%A6%E4%B8%B2&spm=1001.2101.3001.7020)char buf[MAXSIZE]可调用函数：“fgets(buf, MAXSIZE, stdin);”，其中stdin是表示键盘输入设备的文件指针。

**实验需要预先创建好student.txt和csstudent.txt这两个文件，并在studen.txt文件中输入内容，csstuden.txt保持为空即可**。

实验中注意由于C语言对大小写敏感，需注意文件名保持一致。另外字段分隔符”:”统一用英文状态输入。与程序中保持一致。

#include<stdio.h>

#include<string.h>

int main()

{

FILE \*fp1, \*fp2;

char buf[2000];

char check[] = {"网络工程技术"};

if ((fp1 = fopen("student.txt", "r")) == NULL)

{

printf(" Open Failed!");

return 0;

}

if ((fp2 = fopen("csstudent.txt", "a+")) == NULL)

{

printf(" Open Failed!");

return 0;

}

while (!feof(fp1))//文本结束时退出循环

{

fgets(buf, 2000, fp1);//一行行读入

if (strstr(buf, check))//判断是否属于“网络工程技术”

{

char \*a[6];

char buffer[100];

a[0] = strtok(buf, ":");//切割字符串成五部分

a[1] = strtok(NULL, ":");

a[2] = strtok(NULL, ":");

a[3] = strtok(NULL, ":");

a[4] = strtok(NULL, ":");

a[4][3] = 0; //将换行符去掉

sprintf(buffer, "%s:%s:%s:%s:%s\n", a[1], a[0], a[3], a[4], a[2]);//格式化写入buffer

fputs(buffer, fp2);//将buffer写入文本csstudent

}

}

fclose(fp1);

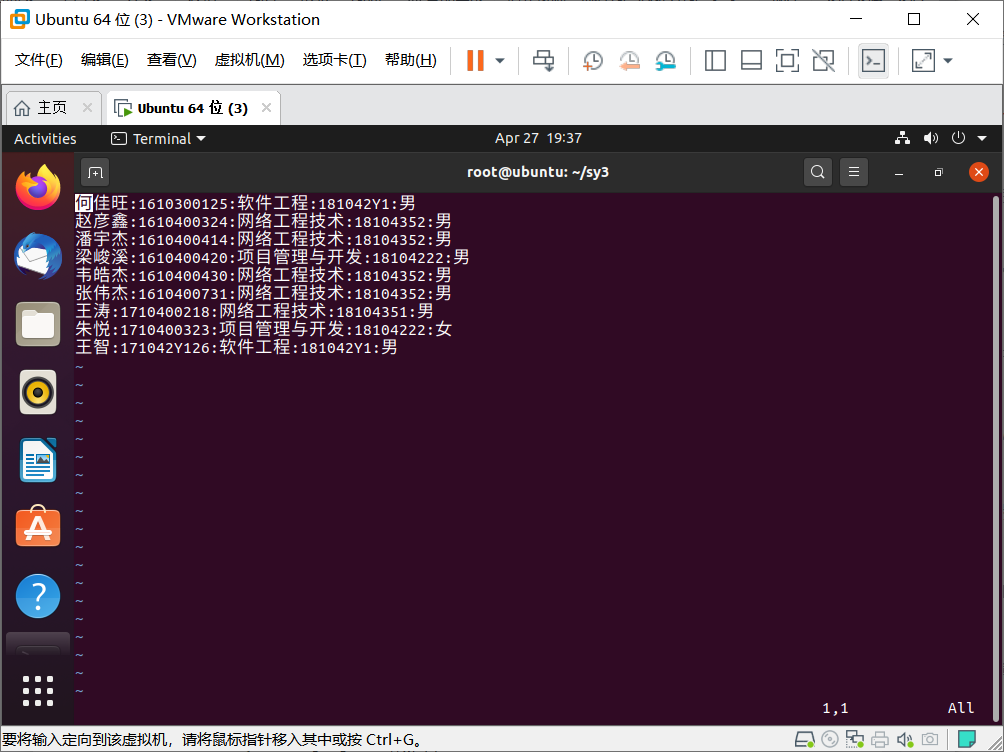
fclose(fp2);

return 0;

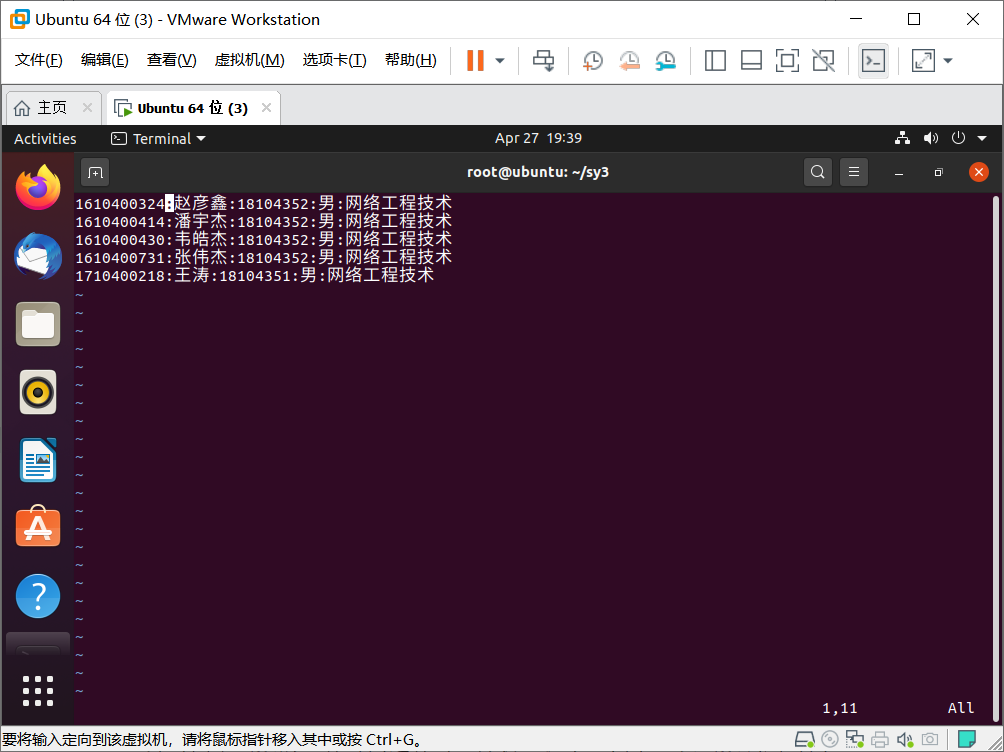
}

运行结果：

student.txt中文件内容示例：



运行后csstudent.txt中文件内容如下：



1. 实验题目
2. 从键盘读入5个学生的成绩信息，包括学号、姓名、语文、数学、英语，成绩允许有一位小数，存入一个结构体数组。

结构体定义为：

typedef struct \_subject {

char sno[20]; //学号

char name[20]; //姓名

float chinese; //语文成绩

float math; //数学成绩

float english; //英语成绩

} subject;

1. 请写一个程序task83.c，测量read、write、fread、fwrite函数调用所需的执行时间，并与prof/gprof工具测的结果进行对比，看是否基本一致。并对四个函数的运行时间进行对比分析。提示：由于一次函数调用时间太短，测量误差太多，应测量上述函数多次（如10000次）运行的时间，结果才会准确。

五、实验报告

将调试运行成功的程序和写好的实验报告一起压缩打包，以实验X-学号-姓名.rar这样的形式命名，并上传提交。