**操作系统课程设计实验报告**

——实验三：文件系统

负责人姓名：黄智勇

学号：14061190

日期：2016.5.21

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 黄智勇 | 14061190 | 实验一 |
| 2 | 王贺 | 14061184 | 实验二 |
| 3 | 兰帅 | 14061203 | 实验三 |
| 4 | 黄智勇 | 14061190 | 实验四 |

目录

[1.实验目的 4](#_Toc446001831)

[2.需求说明 4](#_Toc446001832)

[2.1基本要求 4](#_Toc446001833)

[2.2 提高要求 4](#_Toc446001834)

[2.3 完成情况 4](#_Toc446001835)

[3.设计说明 5](#_Toc446001836)

[3.1 程序流程图 5](#_Toc446001837)

[3.2基本要求实现说明 5](#_Toc446001838)

[3.3 提高要求实现说明 5](#_Toc446001839)

[4.收获和感想 5](#_Toc446001840)

# 1.实验目的

1.了解文件管理系统的作用和工作方式。

2.了解FAT文件系统的结构。

3.学习文件管理系统的一般开发方法。

# 2.需求说明

## 2.1基本要求

准备一个FAT16格式的U盘，在Linux下编写一个文件系统管理程序，对U盘上的文件进行管理。具体要求如下：

1. 设计并实现一个目录列表函数(无须支持选项，如ls -a、ls -l等)，用来显示当前目录下包含的文件信息。
2. 设计并实现一盒改变目录函数(无须处理路径名，如.../../dierctoryName等)，用来把当前目录切换到上一层目录或当前目录的子目录。
3. 设计并实现一个删除文件函数，使用当前目录中的要删除的文件名作为参数，用来删除指定的文件，要注意文件的隐藏、只读和其他系统属性。
4. 设计并实现一个创建文件函数，使用要创建的文件名和文件大小作为参数，用来创建一个新的文件。

要实现的函数：

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | Linux下对应的命令 | 代码实现命令 | 命令格式 |
| ud\_ls | ls | ls | ls |
| ud\_cd | cd | cd | cd.. 切换至上级目录  cd. 切换至当前目录  cd directoryName 切换至当前目录的子目录 |
| ud\_df | rm | df | df fileName |
| ud\_cf | N/A | cf | Cf fileName fileSize |

## 2.2 提高要求

1. 增加创建目录的功能。
2. 增加删除目录的功能：通常需要先判断目录是否为空目录，若目录不为空，则需给出提示，并删除其包含的所有子目录和文件；若是空目录则可以直接删除。
3. 增加绝对路径和多级目录的支持：这里需要对输入的目录路径字符串进行解析，然后逐级查找目录。
4. 对ud\_cd()函数进行改进，使其可以向文件中写入实际内容，并根据写入的内容计算文件的实际大小。
5. 对ud\_ls()函数进行改进，增加对全部非根目录信息的读取（基本要求中仅读取一个扇区的非根目录细信息）

## 2.3 完成情况

基本要求在源代码中就已经基本实现，提高要求也在我们的努力下全部完成了，部分功能有些瑕疵，还需要进一步改进

# 

# 3.设计说明

## 3.1 程序流程图

主控函数：

开始

输入命令

判断输入的命令

ls

df

cf

cd

ud\_cd

ud\_df

ud\_cf

ud\_ls

是否执行结束

否

是

## 

结束

ud\_ls函数：

开始

是否是根目录区

否

是

## 

计算子目录起始地址

显示根目录区

读取目录项信息

## 否

是否读完一个扇区

是

结束

## ud\_df:

开始

查找当前目录

文件是否存在

是

否

在FAT表中依次找到该文件所占用的簇号

把簇号对应的FAT表项的内容标记为未使用

结束

清除对应的目录项

ud\_cf函数：

开始

查找当前目录

是否有同名文件

否

是

遍历FAT表找到空白簇

将簇信息依次写入

在当前目录下为创建的文件建立目录项

结束

## 3.2基本要求实现说明

通读源码后发现基本要求已经实现，故这里不再进行说明。

## 3.3 提高要求实现说明

1. 创建目录：

int fd\_mkdir(char\* name) {

mogic = 1;

char content[120] = "sdjfs";

fd\_cf(name,content);

}

利用这个函数调用文件生成函数fd\_cf函数，在fd\_cf()函数中判断是创建目录还是创建文件。如果要创建目录则执行：

char content[640];

FillNewDir(content,clusterno[0],0); FillNewDir(content,0,32);

WriteContent(clusterno[0],content,64);

FullNewDir()函数如下：

void FillNewDir(char \*content,int currentStartCluster,int startIndex)

{

int i = 0;

if(startIndex==0)

{

content[i+startIndex] = '.';

i++;

}

if(startIndex!=0)

{

content[startIndex] = content[startIndex+1] = '.';

i = 2;

}

for(;i<11;i++)

{

content[startIndex+i] = (char)0x20;

}

if(magic==1)

content[startIndex+11] = (char)0x04;

else

content[startIndex+11] = (char)0x05;

for(i = 12;i<22;i++)

content[startIndex+i] = (char)0x00;

FillTime(content+startIndex+22);

/\*

content[startIndex+22] = (char)(((hour&31)<<3)+((minute>>3)&7));

content[startIndex+23] = (char)(((minute&7)<<5)+(second&31));

content[startIndex+24] = (char)(((year&127)<<1)+((month>>3)&1));

content[startIndex+25] = (char)(((month&7)<<5)+(day&31));

\*/

content[startIndex+27] = (char)((currentStartCluster>>8)&255);

content[startIndex+26] = (char)(currentStartCluster&255);

for(i = 28;i<32;i++)

{

content[startIndex+i] = (char)0x00;

}

}

1. 删除目录：

也是利用删除文件的函数fd\_df()，如果是删除目录就递归调用fd\_df()函数，代码如下：

if(pentry->subdir) {

int bk=magic;

int ret;

struct Entry \*tentry=(struct Entry\*)malloc(sizeof(struct Entry));

fd\_cd(filename);

for(i=curdir->FirstCluster;i!=0xffff;i=GetFatCluster(i)) {

for(j=0;j<CLUSTER\_SIZE;j+=abs(ret)) {

lseek(fd,DATA\_OFFSET+(i-2)\*CLUSTER\_SIZE+j,SEEK\_SET);

ret=GetEntry(tentry);

//printf(tentry->short\_name);putchar('x'); putchar('\n');

if(ret>0 && strcmp(tentry->short\_name,".") && strcmp(tentry->short\_name,"..")) {

puts(tentry->short\_name);

fd\_df(tentry->short\_name);

}

}

}

magic=bk;

fd\_cd("..");

free(tentry);

}

1. 多级目录和根目录的支持：

if(\*dir=='/') {

//根目录

struct Entry\* tmp=curdir;

int pre\_dirno=dirno;

if(dir[1]=='/') {

puts("illegal input");

return -1;

}

curdir=NULL; dirno=0;

ret=fd\_cd(dir+1);

if(ret<0) { //recover on failure

curdir=tmp;

dirno=pre\_dirno;

return ret;

}

return 1;

}

for(pi=dir;\*pi;++pi)

if(\*pi=='/') {

if(pi[1]=='/') {

puts("illegal input");

return -1;

}

\*pi=0; flag=1;

break;

}

多级目录则设置一个for循环，直到查找完所有目录为止。

1. 改进fd\_cf()函数，使其可以写进实际内容，并且可以根据写入内容计算大小，增加Writecontent函数来写内容，计算大小，函数如下：

int WriteContent(int startCluster, char\* content, int length)

{

// Requires: length>=0, startCluster>=2

int start = DATA\_OFFSET+(startCluster-2)\*CLUSTER\_SIZE;

unsigned short cur;

cur = startCluster;

int temp = length;

if(length<0||startCluster<2)

{

return -1;

}

char \*zero=(char\*)malloc(CLUSTER\_SIZE);

memset(zero,0,CLUSTER\_SIZE);

while(temp>0)

{

if(lseek(fd,start,SEEK\_SET)<0)

return -1;

if(temp < CLUSTER\_SIZE)

{

write(fd, content, temp);

content += temp;

lseek(fd,start+temp,SEEK\_SET);

cur=GetFatCluster(cur);

start=DATA\_OFFSET+(cur-2)\*CLUSTER\_SIZE;

}

else

{

write(fd, content, CLUSTER\_SIZE);

content += CLUSTER\_SIZE;

cur = GetFatCluster(cur);

if(cur==0xffff) break;

start = DATA\_OFFSET+(cur-2)\*CLUSTER\_SIZE;

}

temp -= CLUSTER\_SIZE;

}

while(cur!=0xffff) {

printf("%d\n",cur);

lseek(fd,start,SEEK\_SET);

write(fd,zero,CLUSTER\_SIZE);

cur=GetFatCluster(cur);

start=DATA\_OFFSET+(cur-2)\*CLUSTER\_SIZE;

}

return length;

}

1. 改进fd\_ls()函数：修改显示子目录中的函数为：

int current\_cluster;

for(current\_cluster=curdir->FirstCluster;current\_cluster!=0xffff;current\_cluster=RevByte(fatbuf[current\_cluster<<1],fatbuf[current\_cluster<<1|1]))

{

printf("%d\n",current\_cluster);

cluster\_addr = DATA\_OFFSET + (current\_cluster/\*curdir->FirstCluster\*/-2) \* CLUSTER\_SIZE ;

if((ret = lseek(fd,cluster\_addr,SEEK\_SET))<0)

perror("lseek cluster\_addr failed");

offset = cluster\_addr;

/\*只读一簇的内容\*/

while(offset<cluster\_addr +CLUSTER\_SIZE)

{

ret = GetEntry(&entry);

offset += abs(ret);

if(ret > 0)

{

printf("%12s\t"

"%d:%d:%d\t"

"%d:%d:%d \t"

"%d\t"

"%d\t\t"

"%s\n",

entry.short\_name,

entry.year,entry.month,entry.day,

entry.hour,entry.min,entry.sec,

entry.FirstCluster,

entry.size,

(entry.subdir) ? "dir":"file");

}

//printf("%d\n",offset-cluster\_addr);

}

加了一个for循环，一直读取信息直到对全部非根目录信息读取完成。

# 4.收获和感想

这次实验是最后一次OS实验，也是比较难的一次。回顾前三次实验，我们从迷茫和一脸懵逼，到略有头绪。从刚开始分工时的手忙脚乱到现在的有条不紊。我们确实在这门课中收获了许多，对Linux系统下的各种命令调用和C语言的编程更加熟悉，真的是很实用的一门实验课。下面讲一讲我们在这次实验中获得的收获和感想。

第四次实验是文件系统。需要我们编写一个文件管理系统来管理一个FAT16格式的U盘。所谓文件系统，就是指操作系统中负责管理和存储文件信息的软件机构。从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体的说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤消文件等。而FAT文件系统是用来管理小容量的磁盘和简单的目录结构的文件系统，有FAT12、FAT16和FAT32三种类型。我们还了解到了磁盘的组成，包括磁头、磁道、柱面、扇区。同时了解到硬盘容量的计算方式：柱面数\*磁头数\*扇区数\*每扇区字节数。还了解到了FAT文件系统是以簇为单位分配磁盘的，而簇是由若干扇区组成的。FAT16文件系统的结构包括：启动记录、文件分配表、根目录区和数据区。启动记录占用第一个扇区，其逻辑扇区编号为0，包含了硬盘的一系列参数和一段引导程序。文件分配表是一个整数数组，每个整数代表数据区的一个簇号，对于数据区的每个簇，文件分配表的相应入口都包含了编码指明当前簇的状态，文件分配表的逻辑扇区编号为2，且FAT有两个文件分配表以防其中一个被破坏。根目录去的每个入口长度为32字节，目录项的个数固定，每个目录项用来保存一个文件的信息。而数据区用来存储文件和数据，其起始地址位：

512(0x200)+512(0x200)+2\*FAT所占扇区数\*512+512\*32B。

通过我们自己编写一个文件管理系统，我们对于FAT16的文件管理的方式的了解更加深刻，同时对于LINUX系统下的文件管理系统也有了一定的了解。对于课本中文件系统一章里提到的各个概念的含义有了具体的认识，理解得更加透彻。同时我们三个人的团队意识和合作能力进一步提高，分工明确，有条不紊。这次实验我们收获良多，每个人的能力都有一定的提高，获益匪浅。