**黑龙江大学**

**实 验 报 告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | **计算机操作系统** | | | | | |
| **实验项目名称** | **文件系统树型目录** | | | | | |
| **实验时间**  **（日期及节次）** | **2016年10月21日 5-6节** | | | | | |
| **专业** | **计算机** | | **学生所在学院** | | **计算机科学技术学院** | |
| **年级** | **2014级** | | **学号** | | **20123349** | |
| **姓名** | **刘伟** | | **指导教师** | | **朱敬华** | |
| **实验室名称** |  | | | | | |
| **实验成绩** | **预习情况** | **操作技术** | **实验报告** | **附加：综合创新能力** | | **实验**  **综合成绩** |
|  |  |  |  | |  |
| **教师签字** |  | | | | | |

**黑龙江大学教务处**

**实验四 文件系统树型目录**

**一．实验名称：文件系统树型目录**

**二．实验目的：**

利用位示图表示外存的分配情况，新建文件时分配必要的空间，模拟文件分配表记录文件在外存上的存储方式。利用交互式命令实现树型目录结构和文件管理。

**三．实验内容：**

在文件中保存目录内容，创建文件或子目录可以用命令行命令：MD、CD、RD、MK（创建文件）、DEL（删除文件）和DIR。目录项包括文件或目录名称、类型（文件、目录或空目录项）、创建日期以及下一个目录项指针、下一级目录项指针。

磁盘的分配采用链接结构（显式链接）的分配。磁盘空闲存储空间管理采用位示图方法。位示图和显示链接的指针合在一起组成文件分配表。

具体过程：

1、利用位示图模拟外存的空闲空间的情况。

2、创建文件的时候分配空闲的磁盘空间，采用显示链接的方式，利用文件分配表（FAT）记录文件在外存上的存储情况。

3、当删除文件时，回收外存上的空间，修改位示图和文件分配表。

目录管理具体过程：

1、创建初始文件，建立根目录的“.”和“..”目录项。

2、显示命令提示符“$”。

输入命令后根据命令含义完成相应文件操作：

* + MD：在目录文件中创建子目录，同时搜索当前目录最后一个目录项，并保存指针信息；
  + CD：根据当前目录切换到指定目录；
  + RD：搜索所要删除的目录是否为空目录，若是则删除；
  + MK：在当前目录中创建文件名称；
  + DEL：搜索所要删除的文件是否存在，若是则删除；
  + DIR：列出当前目录的所有目录项。

**四．实验代码**

#include<iostream>

#include<time.h>

using namespace std;

struct FCB

{

char name[100];

int year,month,day,hour,minute,second;

int length;

int startaddr;

int node;

}fcb[1000];

struct Directory

{

char name[100][100];

int year,month,day,hour,minute,second;

int judge[100]; /\*0为文件夹,1为文件\*/

int link[100];

int node;

int num;

}Dir[1000];

int path = 1,Time = 1,used\_fcb = 0,used\_dir = 1;

int step = 0;

char steps[100][100]; /\*step保存当前工作路径深度,steps保存当前工作路径\*/

int FAT[256]; /\* FAT表 \*/

int Memory = 0;

char bitmap[32]; /\* 位示图 \*/

time\_t T; /\*用来存系统时间\*/

struct tm \*TimeP; /\*用来存localtime传回的地址\*/

void init(int n,int root) /\*对新结点初始化\*/

{

strcpy(Dir[n].name[0],".");

strcpy(Dir[n].name[1],"..");

Dir[n].judge[0]=Dir[n].judge[1] = 0;

Dir[n].link[0] = Time;

Dir[n].link[1] = root;

Dir[n].node = Time ++;

Dir[n].num = 2;

T = time(NULL);

TimeP = localtime( &T );

Dir[n].year = (TimeP->tm\_year) +1900;

Dir[n].month = (TimeP->tm\_mon) + 1;

Dir[n].day = (TimeP->tm\_mday);

Dir[n].hour = (TimeP->tm\_hour);

Dir[n].minute = (TimeP->tm\_min);

Dir[n].second = (TimeP->tm\_sec);

}

bool get\_bit\_map(int n);

void init\_bit\_map() /\*初始化位示图\*/

{

int i;

for(i = 0 ; i < 32 ; i ++)

bitmap[i] = rand()%256;

for(i = 0 ; i < 256 ; i ++)

FAT[i] = -1;

for(i = 0 ; i < 256 ; i++)

if(get\_bit\_map(i) == 0)

Memory ++;

}

void show\_bit\_map() /\*显示位示图\*/

{

int i,j;

cout << "位示图" <<endl;

for(i = 0 ; i < 16 ; i ++)

{

for(j = 0 ; j < 16 ; j ++)

printf("%d ",bool(bitmap[(i\*16+j)/8] & (1<<(7-(i\*16+j)%8))));

cout<<endl;

}

}

bool get\_bit\_map(int n)

{

return bool(bitmap[n/8] & (1<<(7-n%8)));

}

void set\_bit\_map(int n,int k)

{

if(k)

bitmap[n/8]=bitmap[n/8] | (1<<(7-n%8));

else

bitmap[n/8]=bitmap[n/8] & ~(1<<(7-n%8));

}

void md() /\*创建文件夹\*/

{

int i;

char op[100];

cin>>op;

for(i = 2 ;i < Dir[path-1].num ; i ++)

if(!Dir[path-1].judge[i] && !strcmp(Dir[path-1].name[i],op))

{

cout<<"文件夹已经存在"<<endl;

return ;

}

init(used\_dir++,path);

Dir[path-1].judge[Dir[path-1].num] = 0;

Dir[path-1].link[Dir[path-1].num] = used\_dir;

strcpy(Dir[path-1].name[Dir[path-1].num],op);

Dir[path-1].num++;

}

void rd() /\* 删除文件夹 \*/

{

int i,j;

char op[100];

cin>>op;

for(i = 2 ; i < Dir[path-1].num ; i ++)

if(!Dir[path-1].judge[i] && !strcmp(Dir[path-1].name[i],op))

{

if(Dir[Dir[path-1].link[i]-1].num >2)

cout<<"该文件夹不为空, 不能删除"<<endl;

else

{

for(j = i ; j < Dir[path-1].num-1 ; j ++)

{

Dir[path-1].link[j] = Dir[path-1].link[j+1];

Dir[path-1].judge[j] = Dir[path-1].judge[j+1];

strcpy(Dir[path-1].name[j],Dir[path-1].name[j+1]);

}

Dir[path-1].num--;

cout<<"删除成功"<<endl;

}

return ;

}

cout<<"该文件夹不存在"<<endl;

}

void fat()

{

int i,j,k;

cout << "FAT\n" ;

for(i = 2 ; i < Dir[path-1].num ; i ++)

{

if(Dir[path-1].judge[i])

{

cout << Dir[path-1].name[i] << ": " ;

k = fcb[Dir[path-1].link[i]].startaddr;

for(j=0;j<fcb[Dir[path-1].link[i]].length;j++)

{

cout << k << " ";

k = FAT[k];

}

cout << endl;

}

}

}

void output\_path() /\*输出工作路径\*/

{

cout<<endl<<" C:\\";

for(int i = 0 ; i <= step ; i ++)

{

if(i == step)

break;

else if(i == 0)

cout<<steps[i];

else

cout<<"\\"<<steps[i];

}

}

void dir()

{

int i;

int dir=0,file=0;

int sum=0;

show\_bit\_map();

output\_path();

cout << "的目录" << endl << endl;

for(i = 0 ; i < Dir[path-1].num ; i ++)

if(!Dir[path-1].judge[i])

{

cout<<Dir[Dir[path-1].link[i]-1].year<<"-"<<Dir[Dir[path-1].link[i]-1].month<<"-"<<Dir[Dir[path-1].link[i]-1].day<<"\t"<<Dir[Dir[path-1].link[i]-1].hour<<":"<<Dir[Dir[path-1].link[i]-1].minute<<":"<<Dir[Dir[path-1].link[i]-1].second<<"\t";

cout<<"<DIR>\t";

cout<<Dir[path-1].name[i]/\*<<"\t"<<Dir[Dir[path-1].link[i]-1].node\*/<<endl;

dir ++;

}

else

{

cout<<fcb[Dir[path-1].link[i]].year<<"-"<<fcb[Dir[path-1].link[i]].month<<"-"<<fcb[Dir[path-1].link[i]].day<<"\t"<<fcb[Dir[path-1].link[i]].hour<<":"<<fcb[Dir[path-1].link[i]].minute<<":"<<fcb[Dir[path-1].link[i]].second<<"\t\t";

cout<<fcb[Dir[path-1].link[i]].length<<"字节\t";

cout<<Dir[path-1].name[i]/\*<<"\t"<<fcb[Dir[path-1].link[i]].node\*/<<endl;

sum += fcb[Dir[path-1].link[i]].length;

file ++;

}

cout << "\t\t" << file << "个文件\t\t" << sum << "\t已用字节" << endl;

cout << "\t\t"<< dir << "个目录\t\t" << Memory << "\t可用字节" << endl;

fat();

}

void \_dir(char \*choose)

{

int i,j,k;

for(i = 0 ; i < Dir[path-1].num ; i ++)

{

for(j=k=0;j<=strlen(Dir[path-1].name[i]);j++)

{

if(choose[k]== '\*')

k++;

if(choose[k] != '\*' && choose[k] != '?' && choose[k] == Dir[path-1].name[i][j])

k++;

else if(choose[k] == '?')

k++;

}

if(k>=strlen(choose))

{

if(!Dir[path-1].judge[i])

{

cout<<Dir[Dir[path-1].link[i]-1].year<<"-"<<Dir[Dir[path-1].link[i]-1].month<<"-"<<Dir[Dir[path-1].link[i]-1].day<<"\t"<<Dir[Dir[path-1].link[i]-1].hour<<":"<<Dir[Dir[path-1].link[i]-1].minute<<":"<<Dir[Dir[path-1].link[i]-1].second<<"\t";

cout<<"\t<DIR>\t";

cout<<Dir[path-1].name[i]<<endl;

}

else

{

cout<<fcb[Dir[path-1].link[i]].year<<"-"<<fcb[Dir[path-1].link[i]].month<<"-"<<fcb[Dir[path-1].link[i]].day<<"\t"<<fcb[Dir[path-1].link[i]].hour<<":"<<fcb[Dir[path-1].link[i]].minute<<":"<<fcb[Dir[path-1].link[i]].second<<"\t";

cout<<fcb[Dir[path-1].link[i]].length<<"\t";

cout<<Dir[path-1].name[i]<<endl;

}

}

}

}

bool output[100];

void ls(int p,int deep)

{

int i,j;

for(i = 2 ; i < Dir[p-1].num ; i ++)

{

for(j = 0 ; j < deep ; j ++)

if(output[j] == true)

cout<<"│\t";

else

cout<<"\t";

if(i < Dir[p-1].num-1)

cout<<"├───"<<Dir[p-1].name[i]<<endl;

else

cout<<"└───"<<Dir[p-1].name[i]<<endl;

output[deep] = false;

if(i < Dir[p-1].num-1)

output[deep] = true;

if(!Dir[p-1].judge[i])

ls(Dir[p-1].link[i],deep+1);

}

}

void cd()

{

char op[100];

int i;

cin>>op;

if(!strcmp(op,"\\"))

{

path=1;

step=0;

return ;

}

for(i = 0 ; i < Dir[path-1].num ; i ++)

if(!Dir[path-1].judge[i] && !strcmp(Dir[path-1].name[i],op))

{

path=Dir[path-1].link[i];

if(i == 1 && step)

step--;

else if(i != 1 && i)

{

strcpy(steps[step++],op);

}

return ;

}

cout<<"不存在该文件夹"<<endl;

}

void mk()

{

int i,j,k;

cin>>fcb[used\_fcb].name;

cin>>fcb[used\_fcb].length;

Memory -= fcb[used\_fcb].length;

for(i = 2 ; i < Dir[path-1].num ; i ++)

if(Dir[path-1].judge[i] && !strcmp(Dir[path-1].name[i],fcb[used\_fcb].name))

{

cout<<"文件重名，创建失败"<<endl;

return ;

}

for(i = j = 0 ; i < 256 && j < fcb[used\_fcb].length ; i ++)

if(!get\_bit\_map(i))

j ++;

if(i == 256)

{

cout<<"没有足够空间,不能创建文件"<<endl;

return ;

}

for(i = j = 0 , k = -1 ; i < fcb[used\_fcb].length ; i ++ , j ++)

{

while(get\_bit\_map(j))

j ++;

set\_bit\_map(j,1);

if(k == -1)

fcb[used\_fcb].startaddr = j;

else

FAT[k] = j;

k = j;

}

strcpy(Dir[path-1].name[Dir[path-1].num],fcb[used\_fcb].name);

fcb[used\_fcb].node = Time++;

Dir[path-1].link[Dir[path-1].num] = used\_fcb;

Dir[path-1].judge[Dir[path-1].num] = 1;

T = time(NULL);

TimeP = localtime( &T );

fcb[used\_fcb].year = (TimeP->tm\_year) + 1900;

fcb[used\_fcb].month = (TimeP->tm\_mon) + 1;

fcb[used\_fcb].day = (TimeP->tm\_mday) ;

fcb[used\_fcb].hour = (TimeP->tm\_hour);

fcb[used\_fcb].minute = (TimeP->tm\_min);

fcb[used\_fcb].second = (TimeP->tm\_sec);

Dir[path-1].num++;

used\_fcb++;

}

void del() /\*删除文件\*/

{

int i,j,k;

char na[100];

cin>>na;

for(i = 2 ; i < Dir[path-1].num ; i ++)

if(Dir[path-1].judge[i] && !strcmp(Dir[path-1].name[i],na))

{

k = fcb[Dir[path-1].link[i]].startaddr;

for(j=0;j<fcb[Dir[path-1].link[i]].length;j++)

{

set\_bit\_map(k,0);

k = FAT[k];

Memory ++;

}

for(j = i ; j < Dir[path-1].num-1 ; j ++)

{

Dir[path-1].link[j] = Dir[path-1].link[j+1];

Dir[path-1].judge[j] = Dir[path-1].judge[j+1];

strcpy(Dir[path-1].name[j],Dir[path-1].name[j+1]);

}

Dir[path-1].num--;

cout<<"删除成功"<<endl;

return ;

}

cout<<"没找到文件"<<endl;

}

bool outputpath() /\*输出工作路径\*/

{

cout<<"C:\\>";

for(int i = 0 ; i < step ; i ++)

{

cout<<steps[i]<<'>';

}

return true;

}

void choose()

{

char op[100],choose[100];

while(outputpath() && cin>>op && strcmp(op,"exit"))

{

memset(choose,'\0',sizeof(choose));

if(!strcmp(op,"cd"))

{

cd();

}

else if(!strcmp(op,"rd"))

{

rd();

}

else if(!strcmp(op,"md"))

{

md();

}

else if(!strcmp(op,"ls"))

{

ls(path,0);

}

else if(!strcmp(op,"mk"))

{

mk();

}

else if(!strcmp(op,"del"))

{

del();

}

else if(!strcmp(op,"dir"))

{

getchar();

gets(choose);

if(choose[0]!='\0')

\_dir(choose);

else

dir();

}

else if(!strcmp(op,"cls"))

{

system("cls");

}else if(!strcmp(op,"show")){

show\_bit\_map();

cout<<endl;

fat();

}

}

}

int main()

{

init(0,1);

cout << "MacOS Sierra [版本 10.12]\n" << endl;

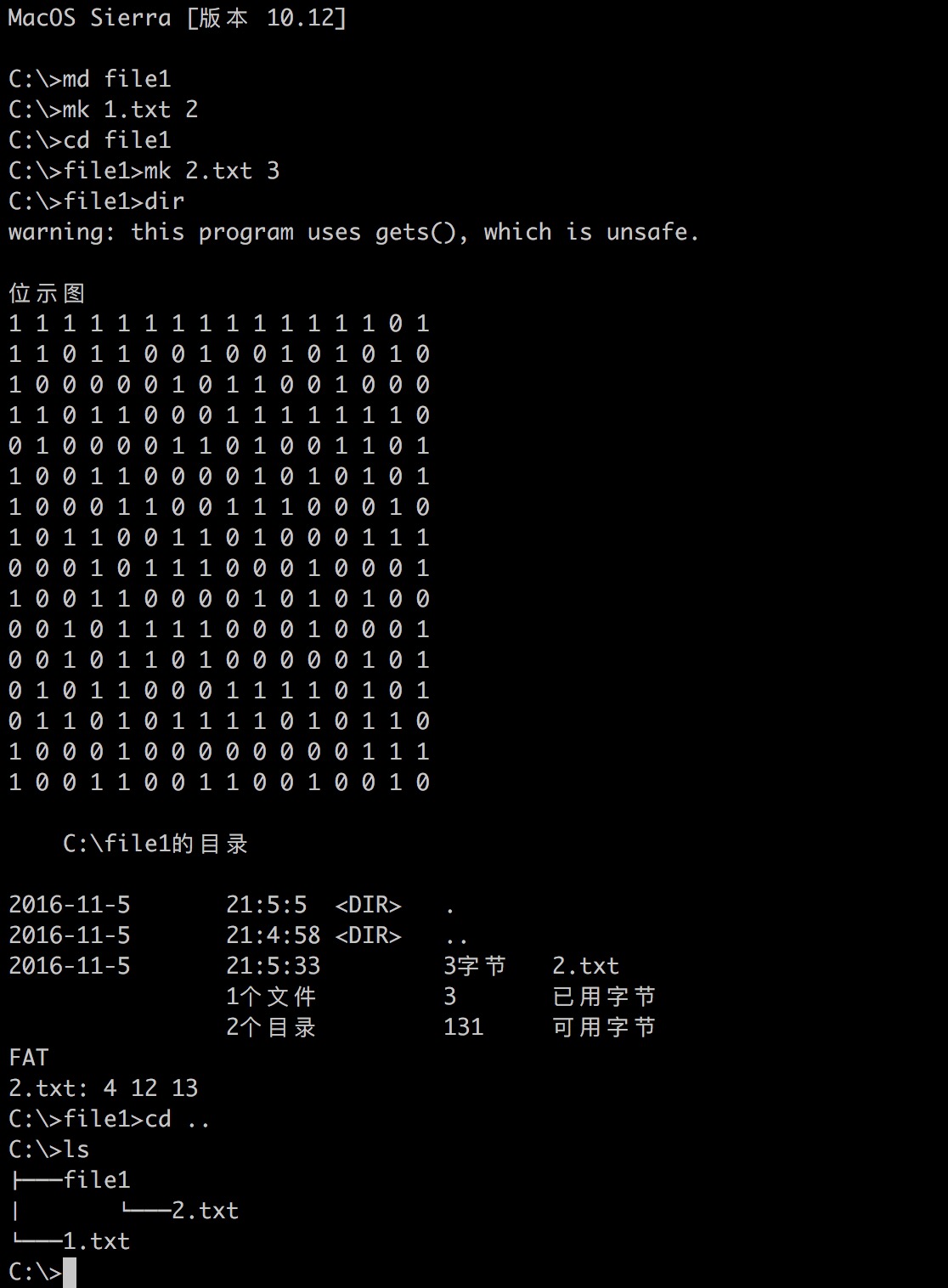
init\_bit\_map();

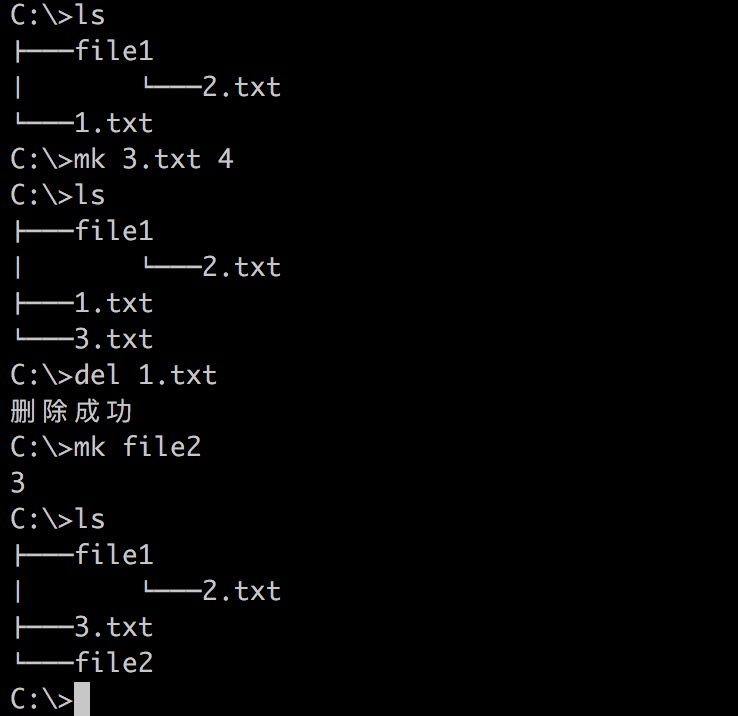
choose();

return 0;

}

**五．实验截图：**

****



**六．实验心得：**

通过本次实验，了解了操作系统文件管理的基本定义和方法。了解了文件系统树型目录创建的基本方式和方法。以及文件创建删除等操作，了解了文件树型结构的基本方法。