#include "stdio.h"

#include "string.h"

#include "stdlib.h"

//目录结构

typedef struct FOLDER{

char name[256]; //目录名

struct FOLDER \*nextFolder; //同级下一目录

struct FOLDER \*frontFolder; //同级上一目录

struct FOLDER \*parentFolder; //父目录

struct FOLDER \*firstChildFolder;//子目录

struct FILE \*firstChildFile; //子文件

int canRead; //是否可读

int canWrite; //是否可写

}FOLDER, \*PFOLDER;

//文件控制块信息

typedef struct FILE{

char name[256]; //文件名

char content[1000]; //文件内容

struct FILE \*frontFile; //同级目录上一文件

struct FILE \*nextFile; //同级目录下一文件

struct FOLDER \*parentFolder; //父目录

int canRead; //是否可读

int canWrite; //是否可写

}FILE, \*PFILE;

PFOLDER root; //根目录

int count = 0; //控制输出格式

int flagD = 0; //删除标志

//查找指定目录

PFOLDER findCurrentFolder(PFOLDER currentFolder, char name[])

{

PFOLDER folder;

if(currentFolder == NULL){

return NULL;

}

if(strcmp(currentFolder->name, name)==0){

return currentFolder;

}

folder = findCurrentFolder(currentFolder->firstChildFolder, name);

if(folder != NULL)

return folder;

folder = findCurrentFolder(currentFolder->nextFolder, name);

if(folder != NULL)

return folder;

return NULL;

}

// 查找指定文件

PFILE findCurrentFile(PFOLDER currentFolder, char name[])

{

PFILE tempFile;

if(currentFolder == NULL)

return NULL;

//遍历当前目录子文件

tempFile = currentFolder->firstChildFile;

while(tempFile != NULL){

//找到该文件

if(strcmp(tempFile->name, name) == 0)

return tempFile;

//当前文件不匹配，则查找当前目录的下一个文件，直到当前目录无文件

tempFile=tempFile->nextFile;

}

//查找上一目录中是否存在该文件

tempFile=findCurrentFile(currentFolder->firstChildFolder, name);

if(tempFile != NULL)

return tempFile;

//查找下一目录中是否存在该文件

tempFile=findCurrentFile(currentFolder->nextFolder, name);

if(tempFile != NULL)

return tempFile;

//查找不到

return NULL;

}

PFOLDER prepareWorkBeforeCreate()

{

PFOLDER currentFolder = NULL;

char name[256];

printf("输入当前目录名称:");

gets(name);

fflush(stdin);

currentFolder=findCurrentFolder(root, name);

if(currentFolder == NULL){

puts("目录不存在！");

return NULL;

}

if(currentFolder->canWrite == 0){

puts("权限不够，不予创建！");

return NULL;

}

return currentFolder;

}

void createFolder()

{

PFOLDER currentFolder=prepareWorkBeforeCreate();

if(currentFolder == NULL){

return;

}

char name[256];

printf("输入新目录的名称：");

gets(name);

fflush(stdin);

PFOLDER newFolder;

newFolder = (PFOLDER)malloc(sizeof(FOLDER));

strcpy(newFolder->name, name);

newFolder->firstChildFolder=NULL;

newFolder->firstChildFile=NULL;

newFolder->nextFolder=NULL;

newFolder->parentFolder=NULL;

newFolder->frontFolder=NULL;

newFolder->canRead=1;

newFolder->canWrite=1;

if(currentFolder->firstChildFolder == NULL){

currentFolder->firstChildFolder=newFolder;

newFolder->parentFolder=currentFolder;

}else{

PFOLDER tempFolder=currentFolder->firstChildFolder;

PFOLDER lastFolder;

while(tempFolder != NULL){

lastFolder=tempFolder;

if(strcmp(tempFolder->name, newFolder->name) == 0){

printf("%s目录下已有同名目录！\n", currentFolder->name);

free(newFolder);

return ;

}

tempFolder=tempFolder->nextFolder;

}

lastFolder->nextFolder=newFolder;

newFolder->frontFolder=lastFolder;

}

puts("创建成功！");

}

void createFile()

{

PFOLDER currentFolder=prepareWorkBeforeCreate();

if(currentFolder == NULL){

return ;

}

char name[256];

printf("请输入新文件名称：");

gets(name);

fflush(stdin);

PFILE newFile;

newFile=(PFILE)malloc(sizeof(FILE));

strcpy(newFile->name, name);

printf("是否输入文件内容？:");

char ans = getchar();

fflush(stdin);

if(ans == 'y' || ans == 'Y'){

printf("输入文件内容：");

gets(newFile->content);

}else{

strcpy(newFile->content, "");

}

fflush(stdin);

newFile->nextFile=NULL;

newFile->frontFile=NULL;

newFile->parentFolder=NULL;

newFile->canRead=1;

newFile->canWrite=1;

if(currentFolder->firstChildFile==NULL){

currentFolder->firstChildFile=newFile;

newFile->parentFolder=currentFolder;

}else{

PFILE tempFile=currentFolder->firstChildFile;

PFILE lastFile;

while(tempFile!=NULL){

lastFile=tempFile;

if(strcmp(tempFile->name, newFile->name) == 0){

printf("%s目录下已经有同名文件！\n", currentFolder->name);

free(newFile);

return;

}

tempFile=tempFile->nextFile;

}

lastFile->nextFile=newFile;

newFile->frontFile=lastFile;

}

puts("创建成功！");

}

void inputName(char name[])

{

printf("输入名称：");

gets(name);

fflush(stdin);

}

//删除该目录下所有内容

void deleteAllChild(PFOLDER currentFolder)

{

PFILE tempFile, dFile;

if(currentFolder==NULL)

return ;

if(flagD)

deleteAllChild(currentFolder->nextFolder);

flagD=1;

deleteAllChild(currentFolder->firstChildFolder);

tempFile=currentFolder->firstChildFile;

while(tempFile!=NULL){

dFile=tempFile;

tempFile=tempFile->nextFile;

free(dFile);

}

free(currentFolder);

}

void deleteFolder()

{

char name[256];

inputName(name);

if(strcmp(name, "root") == 0){

puts("根目录不准删除！");

return;

}

PFOLDER currentFolder=findCurrentFolder(root, name);

if(currentFolder==NULL){

printf("该目录不存在！");

return;

}

if(currentFolder->canWrite==0){

printf("权限不够，无法删除！");

return;

}

if(currentFolder->frontFolder==NULL){

currentFolder->parentFolder->firstChildFolder=currentFolder->nextFolder;

if(currentFolder->nextFolder!=NULL){

currentFolder->nextFolder->frontFolder=NULL;

}

}else{

currentFolder->frontFolder->nextFolder=currentFolder->nextFolder;

}

deleteAllChild(currentFolder);

puts("删除成功！");

}

//删除文件

void deleteFile()

{

char name[256];

inputName(name);

PFILE currentFile=findCurrentFile(root, name);

if(currentFile==NULL){

printf("文件不存在！");

return;

}

if(currentFile->frontFile==NULL){

currentFile->parentFolder->firstChildFile=currentFile->nextFile;

if(currentFile->nextFile!=NULL){

currentFile->nextFile->frontFile=NULL;

}

}else{

currentFile->frontFile->nextFile=currentFile->nextFile;

}

free(currentFile);

puts("删除成功！");

}

void displayFile(PFOLDER currentFolder)

{

PFILE tempFile;

if(currentFolder != NULL && currentFolder->canRead)

{

for(int i=0;i<count;i++){

printf(" ");

}

printf("|-");

printf(currentFolder->name);

int length=15-count\*2-strlen(currentFolder->name);

for(i=0;i<length;i++){

printf(" ");

}

if(count==0)

printf(" canRead canWrite\n");

else

printf("<dir> %d %d\n", currentFolder->canRead,currentFolder->canWrite);

}else if(currentFolder==NULL){

count--;

return;

}

count++;

tempFile=currentFolder->firstChildFile;

while(tempFile!=NULL){

if(tempFile!=NULL){

for(int i =0;i<count;i++){

printf(" ");

}

printf("|-");

printf(tempFile->name);

int length=20-count\*2-strlen(tempFile->name);

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

printf(" ");

if(count==0)

printf(" canRead canWrite\n");

else

printf(" %d %d\n", tempFile->canRead, tempFile->canWrite);

}

tempFile=tempFile->nextFile;

}

displayFile(currentFolder->firstChildFolder);

displayFile(currentFolder->nextFolder);

}

//展示文件内容

void showFileContent()

{

PFILE currentFile=NULL;

char name[256];

printf("请输入文件名：");

gets(name);

fflush(stdin);

currentFile=findCurrentFile(root, name);

if(currentFile==NULL){

puts("文件不存在！");

return ;

}

if(!currentFile->canRead){

puts("权限不够，无法读取！");

return ;

}

printf("文件内容：%s\n", currentFile->content);

printf("文件长度：%d\n", strlen(currentFile->content));

}

void changeFileContent()

{

PFILE currentFile=NULL;

char name[256];

printf("输入文件名：");

gets(name);

fflush(stdin);

currentFile=findCurrentFile(root, name);

if(currentFile==NULL){

puts("文件不存在！");

return;

}

if(!currentFile->canWrite)

{

puts("权限不足，不能修改！");

return ;

}

printf("输入文件内容：");

gets(currentFile->content);

fflush(stdin);

puts("更改成功！");

}

void changeAllChildPermission(PFOLDER currentFolder, int canRead, int canWrite)

{

PFILE tempFile;

if(currentFolder!=NULL)

{

currentFolder->canRead=canRead;

currentFolder->canWrite=canWrite;

}else if(currentFolder==NULL)

return;

tempFile = currentFolder->firstChildFile;

while(tempFile!=NULL){

tempFile->canRead=canRead;

tempFile->canWrite=canWrite;

tempFile = tempFile->nextFile;

}

changeAllChildPermission(currentFolder->firstChildFolder, canRead, canWrite);

if(currentFolder->firstChildFolder!=NULL)

changeAllChildPermission(currentFolder->firstChildFolder->nextFolder, canRead, canWrite);

}

void changeFolderPermission()

{

char name[256];

inputName(name);

if(strcmp(name, "root") == 0){

puts("根目录不允许更改权限！");

return;

}

PFOLDER currentFolder=NULL;

currentFolder=findCurrentFolder(root, name);

if(currentFolder==NULL){

puts("目录不存在！");

return ;

}

printf("输入目录权限（读 写）->如 0 1：");

scanf("%d%d", &currentFolder->canRead, &currentFolder->canWrite);

fflush(stdin);

changeAllChildPermission(currentFolder, currentFolder->canRead, currentFolder->canWrite);

puts("权限修改成功！");

}

void changeFilePermission()

{

char name[256];

inputName(name);

PFILE currentFile=NULL;

currentFile=findCurrentFile(root, name);

if(currentFile==NULL){

puts("文件不存在！");

return ;

}

printf("输入文件权限（读 写）->如 0 1：");

scanf("%d%d", &currentFile->canRead, &currentFile->canWrite);

fflush(stdin);

puts("权限修改成功！");

}

//判断重命名的文件是否重名

int fileNameIsDuplication(PFILE currentFile, char name[])

{

PFILE tempFile = currentFile->frontFile;

while(tempFile!=NULL){

if(strcmp(tempFile->name, name)==0){

printf("文件重名！");

return 1;

}

tempFile = tempFile->frontFile;

}

tempFile = currentFile->nextFile;

while(tempFile!=NULL){

if(strcmp(tempFile->name, name)==0){

printf("文件重名！");

return 1;

}

tempFile = tempFile->nextFile;

}

return 0;

}

//判断目录重命名是否重名

int folderNameIsDuplication(PFOLDER currentFolder, char name[])

{

PFOLDER tempFolder = currentFolder->frontFolder;

while(tempFolder!=NULL){

if(strcmp(tempFolder->name, name)==0){

printf("目录重名！");

return 1;

}

tempFolder = tempFolder->frontFolder;

}

tempFolder = currentFolder->nextFolder;

while(tempFolder!=NULL){

if(strcmp(tempFolder->name, name)==0){

printf("目录重名！");

return 1;

}

tempFolder = tempFolder->nextFolder;

}

return 0;

}

void renameFolder()

{

char name[256];

inputName(name);

if(strcmp("root", name)==0){

puts("根目录无法重命名！");

return ;

}

PFOLDER currentFolder = findCurrentFolder(root, name);

if(currentFolder==NULL){

puts("目录不存在！");

return;

}

printf("输入新名称！");

gets(name);

fflush(stdin);

if(folderNameIsDuplication(currentFolder, name))

return ;

strcpy(currentFolder->name, name);

puts("重命名成功！");

}

void renameFile()

{

char name[256];

inputName(name);

PFILE currentFile=findCurrentFile(root, name);

if(currentFile==NULL){

puts("文件不存在！");

return;

}

printf("输入新名称！");

gets(name);

fflush(stdin);

if(fileNameIsDuplication(currentFile, name))

return;

strcpy(currentFile->name, name);

puts("重命名成功！");

}

void menu()

{

char choice[256];

printf("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\* mkdir: 创建目录 \*\n");

printf("\t\* rmdir: 删除目录 \*\n");

printf("\t\* rnamedir: 重命名目录 \*\n");

printf("\t\* chdir: 更改目录权限 \*\n");

printf("\t\* touch: 创建文件 \*\n");

printf("\t\* rmfile: 删除文件 \*\n");

printf("\t\* rnamefile: 重命名文件 \*\n");

printf("\t\* less: 显示文件内容 \*\n");

printf("\t\* vi: 更改文件内容 \*\n");

printf("\t\* chmod: 更改文件权限 \*\n");

printf("\t\* exit: 退出 \*\n");

printf("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

count = 0;

printf("\n，目录结构：\n");

displayFile(root);

while(1){

printf("\n---------------------------------------------------\n");

printf("请输入操作命令：");

gets(choice);

fflush(stdin);

if(strcmp(choice, "mkdir")==0){

createFolder();

}else if(strcmp(choice, "rmdir")==0){

deleteFolder();

}else if(strcmp(choice, "rnamedir")==0){

renameFolder();

}else if(strcmp(choice, "chdir")==0){

changeFolderPermission();

}else if(strcmp(choice, "touch")==0){

createFile();

}else if(strcmp(choice, "rmfile")==0){

deleteFile();

}else if(strcmp(choice, "rnamefile")==0){

renameFile();

}else if(strcmp(choice, "less")==0){

showFileContent();

}else if(strcmp(choice, "vi")==0){

changeFileContent();

}else if(strcmp(choice, "chmod")==0){

changeFilePermission();

}else if(strcmp(choice, "exit")==0){

exit(0);

}

count = 0;

printf("\n，目录结构：\n");

displayFile(root);

}

}

void initRootFolder()

{

root=(PFOLDER)malloc(sizeof(FOLDER));

root->frontFolder=NULL;

root->nextFolder=NULL;

root->parentFolder=NULL;

root->firstChildFolder=NULL;

root->firstChildFile=NULL;

root->canRead=1;

root->canWrite=1;

strcpy(root->name, "root");

}

int main() {

initRootFolder();

menu();

return 0;

}