//基于多级文件目录的文件管理系统

#include<bits/stdc++.h>

#include <windows.h>

#include <stdio.h>

#include <ctime>

#include <cstdio>

#define endl '\n'

using namespace std;

const int N = 1e4 + 10;

const int M = 110;

struct file{

string name;

vector<string> vet; //当前文件中存储的内容

string where;//用来记录当前文件的位置

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

int \_size;

}b[N];

struct directory{

string name;

vector< pair<string,int> > hold;//记录当前文件存储的文件,pair表示当前的存储是文件还是目录

map<string,int> is\_file; //当前目录下是否存在某个名字的文件

map<string,int> is\_directory; //当前目录下是否存在某个目录

string where;//用来记录当前目录的位置

string pre;//上一个文件，用来回到上一个目录

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

int \_size;

}a[N];

vector< pair<string,int> > fid[N];//存储某个名字所有出现的位置，如果second = 0，表示已经被删除

map<string,int > file\_directory;//找到某个文件或者目录存储的位置

int cnt;//表示每个名字对应的位置

int cnt1, cnt2;//表示存储的文件和目录的位置

int pos = 0;//表示当前所在目录的坐标

char place[N];//表示当前位置

//1表示目录，0表示文件

void manage();

//改变字符显示颜色

void COLOR\_PRINT(const char\* s, int color){

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(handle, FOREGROUND\_INTENSITY | color);

printf(s);

SetConsoleTextAttribute(handle, FOREGROUND\_INTENSITY | 7);

}

void get\_systime(int tmp\_stt, int tmp\_cnt){ //tmp\_cnt表示要进行操作的文件或目录的位置

time\_t rawtime;

struct tm \*ptminfo;

time(&rawtime);

ptminfo = localtime(&rawtime);

if(tmp\_stt == 1){ //为1表示要对目录时间信息进行更新

a[tmp\_cnt].year = ptminfo->tm\_year + 1900;

a[tmp\_cnt].month = ptminfo->tm\_mon + 1;

a[tmp\_cnt].day = ptminfo->tm\_mday;

a[tmp\_cnt].hour = ptminfo->tm\_hour;

a[tmp\_cnt].minute = ptminfo->tm\_min;

a[tmp\_cnt].seconds = ptminfo->tm\_sec;

}

else{ //为0表示要对文件时间信息进行更新

b[tmp\_cnt].year = ptminfo->tm\_year + 1900;

b[tmp\_cnt].month = ptminfo->tm\_mon + 1;

b[tmp\_cnt].day = ptminfo->tm\_mday;

b[tmp\_cnt].hour = ptminfo->tm\_hour;

b[tmp\_cnt].minute = ptminfo->tm\_min;

b[tmp\_cnt].seconds = ptminfo->tm\_sec;

}

}

//该函数用来获取目录和文件的大小

int get\_size(int id, int tmp\_cnt){//tmp\_cnt为目录或文件所在存储位置

int sum = 0;

if(id == 1){ //获取目录的大小

sum = a[tmp\_cnt].is\_file.size() + a[tmp\_cnt].is\_directory.size();

}

else{ //获取文件的大小

for(int i = 0; i < b[tmp\_cnt].vet.size(); i++){

sum += b[tmp\_cnt].vet[i].size();

}

}

return sum;

}

//实现目录打开功能

void open(string poss){ //poss表示要打开的目录的名字

if(poss == ".."){ //linux命令行，cd ..表示返回上一级目录

if(pos >= 1) pos --;

else{

cout <<"wrong: wrong return!" << endl;

}

}

else{//判断当前目录中是否含有要打开的目录文件

if(a[pos].is\_directory.find(poss) == a[pos].is\_directory.end()){

cout <<"-bash: cd: " << poss <<" : Not a directory" << endl;

}

else{

pos = a[pos].is\_directory[poss];

}

}

manage();

}

//该函数用来实现新建目录功能

void mkdir(string poss){ //poss表示所要新建的目录的名字

if(a[pos].is\_directory.find(poss) != a[pos].is\_directory.end()){

cout <<"warning: duplicate directories!"<< endl;

}

else{

cnt1 ++;

string tmp\_pos = a[pos].where;

tmp\_pos += "/";

tmp\_pos += a[pos].name;

if(file\_directory.find(poss) == file\_directory.end()){

cnt ++;

file\_directory[poss] = cnt;

fid[cnt].push\_back({tmp\_pos,1});

}

else{

int x = file\_directory[poss];

fid[cnt].push\_back({tmp\_pos,1});

}

a[cnt1].name = poss;

a[cnt1].where = tmp\_pos;

get\_systime(1,cnt1); //获取新建目录时的系统时间

a[pos].hold.push\_back({poss,1});

a[pos].is\_directory[poss] = cnt1;

a[cnt1].pre = a[pos].name;

}

manage();

}

//该函数用来实现新建文件功能

void touch(string name, int stt){ //name表示所要新建的文件名

string tmp\_pos = a[pos].where;

tmp\_pos += "/";

tmp\_pos += a[pos].name;

if(file\_directory.find(name) == file\_directory.end()){//查找和判断

cnt ++;

file\_directory[name] = cnt;

fid[cnt].push\_back({tmp\_pos,1});

}

else{

int x = file\_directory[name];

fid[x].push\_back({tmp\_pos,1});

}

if(a[pos].is\_file.find(name) != a[pos].is\_file.end()){

cout <<"warning: duplicate files!" << endl;

manage();

}

cnt2 ++;//更新相应数据结构

b[cnt2].where =tmp\_pos;

b[cnt2].name = name;

get\_systime(0,cnt2); //获取新建文件时的系统时间

a[pos].hold.push\_back({name,0});

a[pos].is\_file[name] = cnt2;

if(stt == 1)

manage();

else return;

}

void echo(string st){//st表示想要写到文件中的信息

string s;

s.clear();

string postion;

postion.clear();

int i = 0;

for(i = 1; i < st.size(); i++){

if(st[i] == '"'){

break;

}

s += st[i];

}

for(i = i + 2; i < st.size(); i++){

postion += st[i];

}

if(a[pos].is\_file.find(postion) == a[pos].is\_file.end()){

//查找所想要写入的文件是否存在，不存在进行创建文件和写入

if(a[pos].is\_directory.find(postion) == a[pos].is\_directory.end()){

touch(postion, 0);

b[cnt2].vet.push\_back(s);

cout << b[cnt2].\_size << endl;

}

else{

cout <<"wrong: Illegal naming!" << endl;

}

}

else{ //存在的话，直接查询文件位置，然后进行信息写入即可

int x = a[pos].is\_file[postion];

b[x].vet.push\_back(s);

}

manage();

}

//该函数实现正向读取文件内容

void cat(string poss){ //poss表示

if(a[pos].is\_file.find(poss) == a[pos].is\_file.end()){ //如果找不到文件就报错

cout <<"cat " << poss <<": No such file or directory" << endl;

}

else{//找到文件直接正向读取文件信息即可

int x = a[pos].is\_file[poss];

for(int i = 0; i < b[x].vet.size(); i++){

cout << b[x].vet[i] << endl;

}

}

manage();

}

//该函数实现反向读取文件内容

void tac(string poss){

if(a[pos].is\_file.find(poss) == a[pos].is\_file.end()){

cout <<"cat " << poss <<": No such file or directory" << endl;

}

else{

int x = a[pos].is\_file[poss];

for(int i = b[x].vet.size() - 1; i >= 0; i--){

cout << b[x].vet[i] << endl;

}

}

manage();

}

void show(string order){ //order表示展示命令

int tmp\_tot = 0;

if(order == "all"){ //如果展示命令为all，则展示粗略包含信息

for(int i = 0; i < a[pos].hold.size(); i++){

string st = a[pos].hold[i].first;

char tmp[M];

memset(tmp, 0, sizeof tmp);

for(int j = 0; j < st.size(); j++){

tmp[j] = st[j];

}

if(a[pos].hold[i].second == 0){

COLOR\_PRINT(tmp,7);

tmp\_tot ++;

cout <<" ";

}

else{

COLOR\_PRINT(tmp,1);

tmp\_tot ++;

cout <<" ";

}

}

if(tmp\_tot == 0) cout <<"warning:is null";

cout << endl;

}

else{ //如果为-l,则展示详细信息，包括建立时间，大小和名称

for(int i = 0; i < a[pos].hold.size(); i++){

string st = a[pos].hold[i].first;

char tmp[M];

memset(tmp, 0, sizeof tmp);

for(int j = 0; j < st.size(); j++){

tmp[j] = st[j];

}

int tmp\_pos;

if(a[pos].hold[i].second == 1){

if(a[pos].is\_directory.find(st) != a[pos].is\_directory.end()) tmp\_pos = a[pos].is\_directory[st];

else continue;

if(tmp\_tot == 0){ //输出目录的相关信息

printf("create time\t\tsize\tname\n");

}

printf("%02d-%02d-%02d %02d:%02d:%02d\t",a[tmp\_pos].year,a[tmp\_pos].month, a[tmp\_pos].day,

a[tmp\_pos].hour, a[tmp\_pos].minute,a[tmp\_pos].seconds);

cout << get\_size(1,tmp\_pos)<<'\t';

COLOR\_PRINT(tmp,7);

tmp\_tot ++;

cout << endl;

}

else{ //输出文件的相关信息

if(a[pos].is\_file.find(st) != a[pos].is\_file.end()) tmp\_pos = a[pos].is\_file[st];

else continue;

if(tmp\_tot == 0){

printf("create time\t\tsize\tname\n");

}

printf("%02d-%02d-%02d %02d:%02d:%02d\t",b[tmp\_pos].year,b[tmp\_pos].month, b[tmp\_pos].day,

b[tmp\_pos].hour, b[tmp\_pos].minute,b[tmp\_pos].seconds);

cout << get\_size(0,tmp\_pos)<<'\t';

COLOR\_PRINT(tmp,1);

tmp\_tot ++;

cout << endl;

}

}

if(tmp\_tot == 0) cout <<"warning:is null" << endl;

}

manage();

}

void find\_where(string name){ //寻找名称为name的文件和目录

int tmp\_tot = 0;

if(file\_directory.find(name) == file\_directory.end()){ //如果找不到，输出提示信息

cout << "can't find!" << endl;

}

else{//找到了，遍历进行输出

int x = file\_directory[name];

for(int i = 0; i < fid[x].size(); i++){

if(fid[x][i].second == 1){

cout << fid[x][i].first << endl;

tmp\_tot ++;

}

}

//找不到，输出错误提示信息

if(tmp\_tot == 0) cout <<"can't find!" << endl;

}

manage();

}

void manage(){

string order1,order2;

COLOR\_PRINT("/home/user@LAPTOP-MFH2B9V0:",2);

string tmp\_st = a[pos].where;

tmp\_st +="/";

tmp\_st += a[pos].name;

memset(place, 0, sizeof place);

for(int i = 0; i < tmp\_st.size(); i++){

place[i] = tmp\_st[i];

}

COLOR\_PRINT(place,1);

cout << "$ ";

cin >> order1 >> order2;

if(order1 == "mkdir"){ //新建目录指令

mkdir(order2);

}

else if(order1 == "touch"){//新建文件指令

touch(order2, 1);

}

else if(order1 == "cd"){//打开目录/返回上一级目录

open(order2);

}

else if(order1 == "echo"){//写入信息到文件中去

echo(order2);

}

else if(order1 == "cat"){//正向读取文件

cat(order2);

}

else if(order1 == "tac"){//反向读取文件

tac(order2);

}

else if(order1 == "ls"){//展示目录中所含的文件、子目录信息

show(order2);

}

else if(order1 == "find"){//按名查找

find\_where(order2);

}

else{

cout <<"command '" << order1 <<"' not found" << endl; //指令不存在报错

}

manage();

}

signed main(){

//一些初始化操作

cout <<"Welcome to OS\_work Page (Design Based on Linux)" << endl;

cout <<endl;

a[0].name = "user";

file\_directory["~/home"] = 0;

a[0].where = "~/home";

place[0] = '~';

manage();

return 0;

}