#include <fstream>

#include <iostream>

#include <experimental/filesystem>

#include <string>

#include <iterator>

#include <chrono>

#include <set>

#include <map>

#include <unordered\_map>

#include "cxxopts.hpp"

namespace fs=std::experimental::filesystem;

using namespace std;

typedef map<string,set<string>> myMap; //oui mais pour le partiel

// typedef unordered\_map<string,set<string>> myMap; //+ rapide

//Ligne de compilation qui fonctionne sous linux gcc 6.3 et windows gcc 7.3

//Attention, il y a un bug sous gcc 8.0 + ! Et filesystem ne compile pas

//g++ -Wall index.cpp -o idx -std=c++17 -lstdc++fs

void affHelp()

{

cout<<"Informations sur le programme. "<<endl

<<"Options de lancement :"<<endl

<<"\t -s,--search <filename> recherche exacte sur le nom du fichier"<<endl

<<"\t -p,--partial <filename> recherche partielle sur le nom du fichier (si cela commence)"<<endl

<<"\t -c,--contains <filename> recherche partielle sur le nom du fichier (si cela contient)"<<endl

<<"\t -d,--directory <directory> préciser un répertoire de lancement"<<endl

<<"\t -v,--verbose cette aide."<<endl;

}

void manageOpts(cxxopts::Options& options)

{

options.add\_options()

("s,search","Recherche extacte",cxxopts::value<std::string>())

("p,partial","Recherche partielle (commence)",cxxopts::value<std::string>())

("c,contains","Recherche partielle (contient)",cxxopts::value<std::string>())

("v,verbose","Informations fonctionnement")

("h,help","Informations fonctionnement")

("d,directory","Répertoire de recherche",cxxopts::value<std::string>())

;

}

bool search (string mot,myMap& table\_hash)

{

bool res=false;

table\_hash.find(mot) == table\_hash.end() ? res= false:res=true;

return res;

}

void pathToTable(myMap& table\_hash,fs::path dir)

{

set <string> mySet;

string path\_dir,path\_file;

for(auto& p: fs::recursive\_directory\_iterator(dir))

{

path\_file=p.path().filename().string();

//path\_dir=p.path().parent\_path().string();

path\_dir=p.path().string(); //changer si on veut tout le chemin

if (search(path\_file,table\_hash))

{

mySet=table\_hash.find(path\_file)->second;

mySet.insert(path\_dir);

table\_hash.at(path\_file)=mySet;

}

else

{

mySet.clear();

mySet.insert(path\_dir);

table\_hash.insert({path\_file,mySet});

}

}

}

vector<string> findSearch(string fichier,myMap& table\_hash)//sort un seul résultat

{

vector<string> result;

if (search(fichier,table\_hash))

{

set<string>::iterator itr;

set<string> chemins=table\_hash.find(fichier)->second;

for (itr = chemins.begin(); itr != chemins.end(); ++itr) //affichage du set

//cout << '\t' << \*itr<<endl;

result.push\_back(\*itr);

// cout<<endl;

}

return result;

}

vector<string> findPartialC(string fichier,myMap& table\_hash) //Bonjour je parcours l'intégralité de la map et affiche si ça contient

{

vector<string> result;

myMap::iterator i;

set<string>::iterator itr;

for(i=table\_hash.begin(); i!=table\_hash.end(); ++i)

{

if ((i->first).find(fichier) != string::npos)

{

for (itr = i->second.begin(); itr != i->second.end(); ++itr) //affichage du set

// cout << '\t' << \*itr;

result.push\_back(\*itr);

// cout<<endl;

}

}

return result;

}

vector<string> findPartialS(string fichier,myMap& table\_hash)//ça part du truc trouvé et ça continue tant que le début commence par ça

{

vector<string> result;

myMap::iterator i;

set<string>::iterator itr;

for(i=table\_hash.lower\_bound(fichier);((i->first).find(fichier) != string::npos); ++i)

{

for (itr = i->second.begin(); itr != i->second.end(); ++itr) //affichage du set

// cout << '\t' << \*itr;

result.push\_back(\*itr);

// cout<<endl;

}

return result;

}

int main(int argc,char\*\* argv)

{

cxxopts::Options options("Prog recherche", "Programme d'indexation d'une arborescence");

manageOpts(options);

auto result = options.parse(argc, argv);

if ((result["v"].as<bool>()) or (result["h"].as<bool>()))

affHelp();

else

{

myMap table\_hash;

chrono::steady\_clock::time\_point t1;

chrono::steady\_clock::time\_point t2;

std::string dossier;

std::string fichier;

char recherche='s';

if (result.count("d")>0)

dossier=result["d"].as<std::string>();

else

dossier=fs::current\_path().string();

if (result.count("s")>0)

{

fichier=result["s"].as<std::string>();

recherche='s';

}

else if (result.count("p")>0)

{

fichier=result["p"].as<std::string>();

recherche='p';

}

else if (result.count("c")>0)

{

fichier=result["c"].as<std::string>();

recherche='c';

}

else return 1;

fs::path dir=fs::path(dossier);

t1 = chrono::steady\_clock::now();

pathToTable(table\_hash,dir);

t2 = chrono::steady\_clock::now();

cout<<"Fin de la recherche"<<endl;

chrono::duration<double> elapsed=chrono::duration\_cast<chrono::duration<double>>(t2-t1);

cout<<"Temps de l'indexation de l'arborescence: "<<elapsed.count()<<" secondes"<<endl;

vector<string> result;

if (!table\_hash.empty()){

cout<<"Méthode : ";

switch (recherche) {

case 's':

cout<<"Recherche totale"<<endl;

t1 = chrono::steady\_clock::now();

result=findSearch(fichier,table\_hash);

t2 = chrono::steady\_clock::now();

break;

case 'p':

cout<<"Recherche partielle"<<endl;

t1 = chrono::steady\_clock::now();

result=findPartialS(fichier,table\_hash);

t2 = chrono::steady\_clock::now();

break;

case 'c':

cout<<"Recherche partielle"<<endl;

t1 = chrono::steady\_clock::now();

result=findPartialC(fichier,table\_hash);

t2 = chrono::steady\_clock::now();

break;

}

if (result.empty())

cout<<"Aucun résultat trouvé"<<endl;

else

{

chrono::duration<double> elapsed=chrono::duration\_cast<chrono::duration<double>>(t2-t1);

cout<<"Temps de la recherche: "<<elapsed.count()<<" secondes"<<endl;

cout<<"Chemins du fichier '"<<fichier<<"' avec la méthode choisie."<<endl;

for(string n : result)

cout<<n<<endl;

}

}

else

cout<<"La table est vide"<<endl;

}

return (0);

}