#include <unistd.h> **// Biblioteca: manejo de directorios y archivos.**

#include <stdio.h> **// Biblioteca: funciones de entrada y salida.**

#include <stdlib.h> **// Biblioteca: gestión de memoria y procesos.**

#include <sys/types.h> **// Biblioteca: búsqueda, orden y manipulación de archivos y directorios.**

#include <sys/stat.h> **// Biblioteca: metadatos de archivos**.

#include <fcntl.h> **// Biblioteca: operaciones de control sobre el descriptor del archivo.**

#include <errno.h> **// Biblioteca: informes de errores.**

#include <string.h> **// Biblioteca: operaciones con cadenas de caracteres**.

#include <stdbool.h> **// Biblioteca: tipo dato bool.**

#include <dirent.h> **// Biblioteca: manejo de carpetas.**

#include <ftw.h> **// Biblioteca: recorrido en árbol de directorios.**

#include <signal.h> **// Biblioteca: manejo de señales.**

#include <limits.h> **// Biblioteca: rango de variables**.

**// Función que recibe un archivo como parámetro y devuelve su tamaño.**

int size\_file (char \* file) ;

**// Función que recibe un directorio como parámetro y devuelve los elementos que contiene.**

int length\_dir (char \* dir) ;

**// Función que recibe un directorio y me devuelve la lista de archivos.**

char \*\* list\_file (char \* dir, bool full) ;

**// Función que comprueba si un fichero es un directorio.**

bool is\_directory (char \* file) ;

**// Función que realiza una búsqueda sobre una carpeta.**

bool search (char \* file, char \* dir) ;

**// Función que busca un fichero en un sistema de carpetas.**

bool recursive\_search (char \* file, char \* dir) ;

**// Función que cuenta el número de veces que aparece un archivo en un sistema de carpetas.**

int recursive\_length\_dir (char \* dir ) ;

**// Función que cuenta el número de carpetas dentro de una carpeta.**

int number\_dir (char \* dir) ;

**// Función que realiza la lista de carpetas de una carpeta.**

char \*\* list\_dir (char \* dir, bool full) ;

int main(int argc, char \*argv[]) {

if ( recursive\_search (argv[1], argv[2]) )

printf ( "Encontrado " ) ;

return (EXIT\_SUCCESS) ;

}

**// Función que recibe un archivo como parámetro y devuelve su tamaño.**

int size\_file (char \* file) {

**// DECLARACIÓN DE VARIABLES**

struct stat metadatos ;

int size = -1 ;

// Asignación de metadatos.

int metadatos\_true = stat ( file , & metadatos ) ;

// Comprobación de metadatos.

if (metadatos\_true >= 0) {

size = metadatos.st\_size ;

}

**// SALIDA DE DATO**

return (size) ;

}

**// Función que recibe un directorio como parámetro y devuelve los elementos que contiene.**

int length\_dir (char \* dir) {

**// DECLARACIÓN DE VARIABLES**

DIR \* directorio = NULL ;

int length = -1 ;

struct dirent \* estructura ;

// Asignación del flujo al directorio.

directorio = opendir ( dir ) ;

**// Comprobación de carpeta.**

if (directorio != NULL) {

length = 0 ;

// CÁLCULOS

while ( estructura = readdir ( directorio ) ) {

if ( strcmp ( estructura->d\_name , "." ) ) {

if ( strcmp ( estructura->d\_name , ".." ) ) {

length++ ;

}

}

}

}

**// SALIDA DE DATOS**

return ( length ) ;

}

**// Función que recibe un directorio y me devuelve la lista de archivos.**

char \*\* list\_file (char \* dir, bool full) {

// DECLARACIÓN DE VARIABLES

char \*\* list ;

const int LENGTH = length\_dir (dir) ;

DIR \* directorio = NULL ;

struct dirent \* estructura ;

int i = 0 ;

**// Asignación del flujo al directorio.**

directorio = opendir ( dir ) ;

if (directorio != NULL) {

// Asignamos el numero de filas deseadas.

list = (char \*\*) malloc (LENGTH \* sizeof (char \*)) ;

while ( estructura = readdir ( directorio ) ) {

if ( strcmp ( estructura->d\_name , "." ) ) {

if ( strcmp ( estructura->d\_name , ".." ) ) {

if (!full) {

int size = strlen(estructura->d\_name) ;

list[i] = (char \*) malloc (size \* sizeof(char)) ;

strcpy (list[i], estructura->d\_name) ;

}

else {

int size = strlen(dir) + sizeof("/") + strlen(estructura->d\_name) ;

list[i] = (char \*) malloc (size \* sizeof(char)) ;

strcpy (list[i], dir) ;

strcat (list[i], "/") ;

strcat (list[i], estructura->d\_name) ;

}

i++ ;

}

}

} **// Fin del while**

} **// Fin de comprobación de directorio.**

return (list) ;

}

**// Función que comprueba si un fichero es un directorio.**

bool is\_directory (char \* file) {

struct stat metadatos ;

if ( lstat(file, & metadatos) < 0 ) {

return (false) ;

}

if ( S\_ISDIR(metadatos.st\_mode) ) {

return (true) ;

}

return (false) ;

}

**// Función que realiza una búsqueda sobre una carpeta.**

bool search (char \* file, char \* dir) {

**// Comprobación de que es una carpeta.**

if (!is\_directory(dir)) {

return (false) ;

}

**// DECLARACIÓN DE VARIABLES**

const int LENGTH = length\_dir (dir) ;

char \*\* list = list\_file (dir,false) ;

bool existe = false ;

**// CÁLCULOS Y SALIDA DE DATOS**

for (int i = 0 ; i < LENGTH && !existe ; i++) {

printf("comprobando: %s\n", list[i]);

if ( strcmp (file, list[i]) == 0 ) {

existe = true ;

}

}

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

free (list[i]) ;

}

free (list) ;

return (existe) ;

}

**// Función que busca un fichero en un sistema de carpetas.**

bool recursive\_search (char \* file, char \* dir ) {

// Comprobación de que es una carpeta.

if (!is\_directory(dir) || length\_dir(dir) <= 0) {

return (false) ;

}

**// Comprobación de que esta el archivo en la carpeta actual.**

if ( search (file, dir) ) {

return (true) ;

}

**// DECLARACIÓN DE CONSTANTES**

const int LENGTH = number\_dir (dir) ;

printf("%s\n", dir);

**// DECLARACIÓN DE VARIABLES**

char \*\* list\_dir\_long = list\_dir (dir,true) ;

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

if ( recursive\_search (file, list\_dir\_long[i]) ) {

return (true) ;

}

}

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

free (list\_dir\_long[i]) ;

}

free (list\_dir\_long) ;

return (false) ;

}

**// Función que cuenta el número de veces que aparece un archivo en un sistema de carpetas.**

int recursive\_length\_dir (char \* dir ) {

// Comprobación de que es una carpeta.

if (!is\_directory(dir)) {

return (false) ;

}

**// DECLARACIÓN DE VARIABLES**

const int LENGTH = length\_dir (dir) ;

char \*\* list = list\_file (dir,true) ;

int occurrence = 0 ;

char \* new\_p\_dir = NULL ;

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

if ( is\_directory(list[i]) ) {

occurrence += recursive\_length\_dir (list[i]) ;

}

else {

occurrence++ ;

}

}

return (occurrence) ;

}

**// Función que cuenta el número de carpetas dentro de una carpeta.**

int number\_dir (char \* dir) {

// DECLARACIÓN DE VARIABLES

DIR \* directorio = NULL ;

int length = -1 ;

struct dirent \* estructura ;

**// Asignación del flujo al directorio.**

directorio = opendir ( dir ) ;

**// Comprobación de carpeta.**

if (directorio != NULL) {

length = 0 ;

**// CÁLCULOS**

while ( estructura = readdir ( directorio ) ) {

if ( strcmp ( estructura->d\_name , "." ) ) {

if ( strcmp ( estructura->d\_name , ".." ) ) {

int size = strlen(dir) + sizeof("/") + strlen(estructura->d\_name) ;

char \* new\_p\_dir = (char \*) malloc (size \* sizeof(char)) ;

strcpy (new\_p\_dir, dir) ;

strcat (new\_p\_dir, "/") ;

strcat (new\_p\_dir, estructura->d\_name) ;

if ( is\_directory(new\_p\_dir) ) {

length++ ;

}

}

}

}

}

**// SALIDA DE DATOS**

return ( length ) ;

}

**// Función que realiza la lista de carpetas de una carpeta.**

char \*\* list\_dir (char \* dir, bool full) {

**// DECLARACIÓN DE VARIABLES**

char \*\* list ;

const int LENGTH = length\_dir (dir) ;

DIR \* directorio = NULL ;

struct dirent \* estructura ;

int i = 0 ;

**// Asignación del flujo al directorio.**

directorio = opendir ( dir ) ;

if (directorio != NULL) {

**// Asignamos el número de filas deseadas.**

list = (char \*\*) malloc (LENGTH \* sizeof (char \*)) ;

while ( estructura = readdir ( directorio ) ) {

if ( strcmp ( estructura->d\_name , "." ) ) {

if ( strcmp ( estructura->d\_name , ".." ) ) {

int size = strlen(dir) + sizeof("/") + strlen(estructura->d\_name) ;

char \* new\_p\_dir = (char \*) malloc (size \* sizeof(char)) ;

strcpy (new\_p\_dir, dir) ;

strcat (new\_p\_dir, "/") ;

strcat (new\_p\_dir, estructura->d\_name) ;

if (is\_directory(new\_p\_dir)) {

if (!full) {

int size = strlen(estructura->d\_name) ;

list[i] = (char \*) malloc (size \* sizeof(char)) ;

strcpy (list[i], estructura->d\_name) ;

}

else {

int size = strlen(dir) + sizeof("/") + strlen(estructura->d\_name) ;

list[i] = (char \*) malloc (size \* sizeof(char)) ;

strcpy (list[i], dir) ;

strcat (list[i], "/") ;

strcat (list[i], estructura->d\_name) ;

}

i++ ;

}

}

}

} **// Fin del while**

} **// Fin de comprobación de directorio.**

return (list) ;

}