**lsoB-t04-g08**



**Juan David Moran Santiusty**

**Andrés Felipe Sandino Gomez**

**Taller número 4 en el curso Laboratorio Sistemas Operativos**

**Profesor:**

**PABLO A. MAGÉ I.**

**Universidad del Cauca**

**Facultad de Ingeniería Electrónica y Telecomunicaciones**

**Departamento de Sistemas**

**Laboratorio Sistemas Operativos**

**Popayán, marzo 2025**

**Juan David Moran Santiusty**

**Andrés Felipe Sandino Gomez**

**lsoB-t04-g08**

Taller No. 4 presentado en el curso de Laboratorio Sistemas Operativos

Estudiantes del:

Programa de Ingeniería de Sistemas

Profesor:

Pablo A. Magé I

Popayán

2025

Contenido

[Contenido i](#_Toc128686608)

[Lista de Figuras ii](#_Toc128686609)

[Título del trabajo 1](#_Toc128686612)

[1.- Planteamiento del Problema 1](#_Toc128686613)

2[.- Estructura General y Makefile](#_Toc128686616) 3

3[.- Inicio de programa y validaciones](#_Toc128686616) 5

# Lista de Figuras

Figura 1, Archivo Makefile4

Figura 2.1, Inicio del programa5

Figura 2.2, Segunda validación6

Figura 2.3, es\_directorio6

**lsoB-t04-g08**

# 1.- Planteamiento del Problema

Deberá desarrollar un programa que, mediante llamada a los servicios del sistema y las funciones de la biblioteca estándar de C, permita buscar archivos dentro de una estructura de directorios del sistema. El programa deberá recibir por línea de comandos dos parámetros: El directorio base de la búsqueda, y el patrón (el texto) que debe encontrarse en el nombre del archivo.

Algunas funciones de la biblioteca estándar de C y/o POSIX que se pueden usar son:

• stat

• opendir

• readdir

• malloc

• strstr

• realpath

A continuación, se propone una posible implementación en pseudocódigo de la búsqueda.

//Retorna 1 si la ruta dada es un directorio, 0 en caso contrario

funcion es\_directorio(cadena ruta)

//Usar el servicio stat para obtener la información de la ruta dada

resultado = stat(ruta, &s)

si resultado < 0 // error

perror("stat")

retornar 0 //C no tiene tipo booleano, 0 = falso

fin si

//Usar el macro S\_ISDIR sobre el atributo st\_mode de la estructura

//para verificar si es un directorio

si S\_ISDIR(s.st\_mode)

retornar 1 //C no tiene tipo booleano, 1 = verdadero

sino

retornar 0

fin si

fin funcion

//Busca recursivamente archivos/directorios cuyo nombre contenga patron

//Retorna el total de coincidencias

funcion buscar(cadena directorio, cadena patron)

total = 0

//Abrir el directorio con opendir

d = opendir(directorio)

//Validar el resultado de la llamada al servicio opendir

//Tenga en cuenta! la comparación de igualdad en C es con ==

si d = nulo

perror("opendir")

Carrera 2 No. 15N esquina-Sector Tulcán

Popayán-Cauca-Colombia

Teléfono: 6028209800 ext. 2100 ó 2101.

decafiet@unicauca.edu.co | www.unicauca.edu.co

retornar total

fin si

//Leer el directorio con el servicio readdir

mientras ((ent = readdir(d)) != nulo)

//Construir la ruta completa directorio/ent->d\_name

ruta = directorio + "/" + ent->d\_name

//Tenga en cuenta! En C no se puede concatenar las cadenas de

//esta forma

//Se debe reservar memoria suficiente con malloc, usar strcpy

//para copiar y strcat para concatenar.

//La cadena de destino (ruta) debe tener suficiente espacio

//para almacenar directorio, "/" y ent->d\_name.

//Toda cadena válida en C termina en nulo, por lo tanto se

//debe reservar 1 byte más de memoria

//Verificar si la entrada (archivo o directorio) contiene

//el patrón de búsqueda

si strstr(ent->d\_name, patron) != nulo

//imprimir la ruta completa

imprimir ruta

total = total + 1

fin si

//Si la entrada es un directorio, se debe buscar

//recursivamente!

//Precaucion: descartar los directorios "." (actual)

//y ".." (anterior) para prevenir que la funcion nunca termine

//Tenga en cuenta! En C, las cadenas se comparan como strcmp

si es\_directorio(ruta) && ent->d\_name != "." && ent->d\_name != ".."

//Buscar recursivamente dentro del subdirectorio

total = total + buscar(ruta, patron)

fin si

fin mientras

//Cerrar el directorio

closedir(d)

retornar total

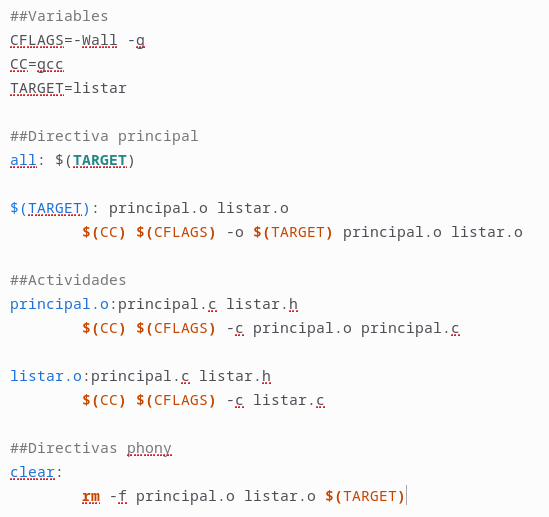
fin funcion

# 2.- Estructura general y Makefile

Para resolver el ejercicio se decidió utilizar una estructura de tres archivos y un archivo Makefile para la compilación de los archivos, la estructura tiene tres archivos.

* principal.c: Donde se encuentra la función main.
* listar.h: Donde se encuentra la declaración de las funciones necesarias.
* listar.c: Donde se encuentra la implementación de las funciones declaradas anteriormente.

La siguiente imagen muestra el archivo Makefile



(Figura 1, Archivo Makefile)

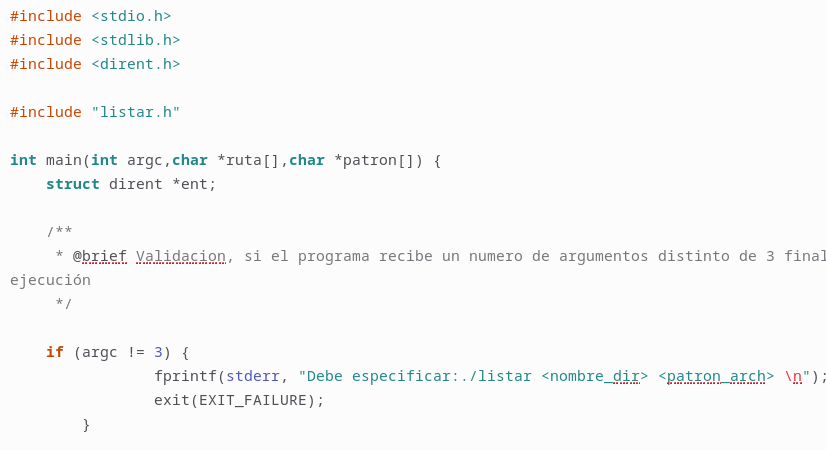
Las partes mas importantes que podemos mencionar serian “CC“ el compilador a utilizar “CFLAGS” instrucciones de compilación donde se agrega “-g” que habilita el depurador y “TARGET” seria el ejecutable.

En la sección de “##Directiva principal” encontramos la instrucción para convertir principal.o y listar.o a objeto.

En la sección de “##Actividades” encontramos las instrucciones para la compilación de principal.c y listar.c, para la compilación de estos archivos también se deben utilizar los archivos que tengan dependencias con estos.

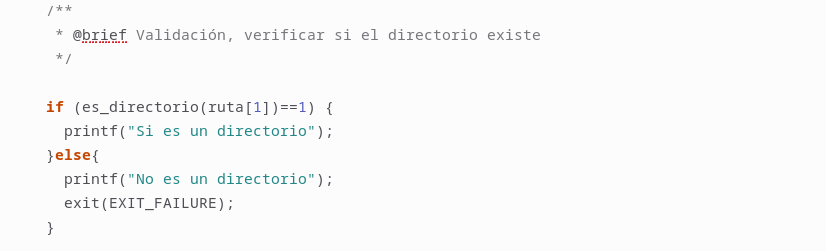
En la sección de “##Directivas phony” encontramos la instrucción para eliminar los archivos .o .

# .3- Inicio de programa y validaciones

(Figura 2.1, Inicio del programa)

La función main tiene tres parámetros iniciales “argc” numero de parametros iniciales “ruta” cadena de texto que almacena la ruta de directorio a buscar y “patron” cadena de texto que guarda el patrón de nombre a buscar en la ruta antes diguitada.

Se tiene una validación inicial la cual verifica que se hayan ingresado los valores requeridos que serian “ruta” y “patron” en caso contrario finalizara la ejecución del programa y mostrara un mensaje de error.

(Figura 2.2, Segunda validación)

Si se supera la primera validación se pasara a verificar si la ruta ingresada es un directorio que existe en el sistema si no existe se finalizara la ejecución y se mostrar un mensaje de error.



(Figura 2.3, es\_directorio)

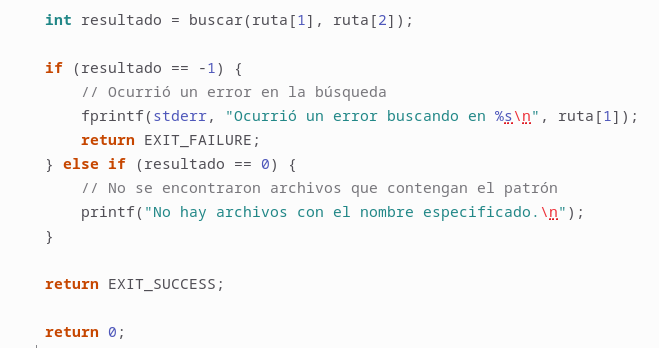
La función “es\_directorio” retorna 1 se la ruta es un directorio de lo contrario se retorna 0, la función recibe el parámetro “ruta” que es la ruta de directorio a buscar y pasa a definir una variable s de tipo “stat” que es una estructura que guarda la información de un directorio.

La función “stat(ruta,&s)” recibe dos parámetros, la ruta a buscar y la estructura donde se guardara su información en caso de que retorne un numero<0 significa que la ruta ingresada es incorrecta y no existe ningún archivo o directorio en la ruta.

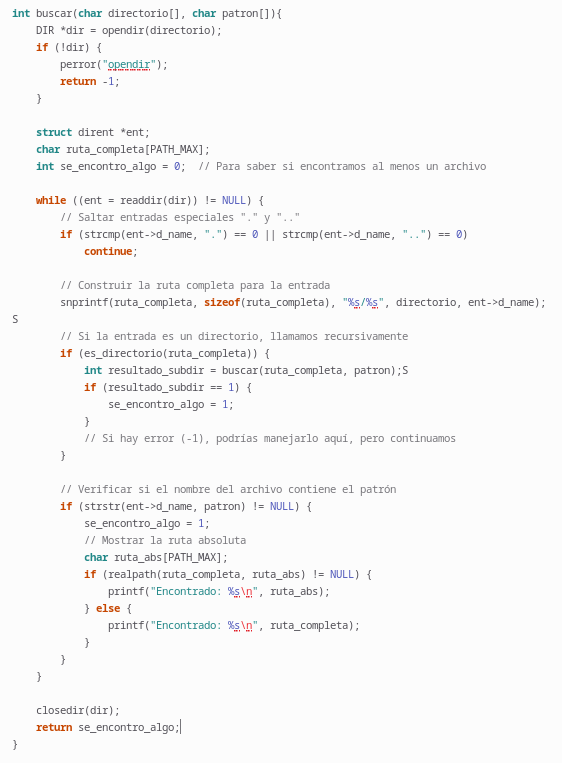
Superada la primera validación se hace uso de la macro S\_ISDIR para verificar si la información guardada en s corresponde a un directorio si es así retorna 1 de lo contrario retorna 0.

Es importante mencionar que para poder usar las funciones stat y S\_ISDIR es necesario importar la librería “<sys/stat.h>”.

En el main, se gestionan los argumentos recibidos por línea de comandos, validando que se haya proporcionado el directorio base y el patrón de búsqueda, y comprobando que el primer argumento realmente corresponde a un directorio válido. Una vez realizadas estas comprobaciones, se invoca la función buscar con los parámetros adecuados. Según el resultado de la búsqueda, el programa informa al usuario si se encontraron archivos que cumplan con el patrón o, en caso contrario, muestra un mensaje indicando que no hay coincidencias.

(Figura 2.4, buscar\_en\_main)

La función buscar ubicada en listar.c recorre recursivamente el directorio base y sus subdirectorios, buscando aquellos archivos cuyo nombre contenga la cadena especificada. Utiliza funciones del sistema como opendir y readdir para acceder a los contenidos de cada directorio y, mediante la función strstr, filtra los archivos que cumplen con el criterio de búsqueda. Cuando encuentra una coincidencia, obtiene y muestra la ruta absoluta del archivo, y si detecta un subdirectorio, vuelve a invocar la función para seguir explorando en profundidad.



(Figura 2.5, buscar)

[1] M. Petersen, K. and Feldt, R. and Mujtaba, S. and Mattsson, “Systematic Mapping Studies in Software Engineering,” *Proc. 12th Int. Conf. Eval. Assess. Softw. Eng.*, pp. 68–77, 2008, doi: 10.1142/S0218194007003112.