



## **Escuela de Ingeniería en Computación**

### **Documentación**

Proyecto #1 Laberinto

### **Curso:**

Sistemas Operativos

### **Estudiantes:**

Kenny Vega Obando 2019162050

Sahid Rojas Chacon 2018319311

### **Profesora:**

Erika Marín Schumann

07 de abril del 2024

<b>Introducción</b>	<b>3</b>
<b>Estrategia de solución</b>	<b>3</b>
<b>Análisis de resultados</b>	<b>4</b>
<b>Lecciones aprendidas</b>	<b>4</b>
<b>Casos de pruebas</b>	<b>5</b>
<b>Comparación con creación de Procesos Fork</b>	<b>9</b>
<b>Manual de usuario</b>	<b>10</b>
<b>Bitácora de trabajo</b>	<b>12</b>
<b>Bibliografía</b>	<b>13</b>

# Introducción

Este documento describe el desarrollo de un programa en C para Linux que simula el recorrido de un laberinto utilizando hilos (threads), el objetivo principal del programa es explorar un laberinto definido en un archivo de texto, utilizando los hilos para explorar todos los caminos y verificar si existe una, ninguna o varias salidas, durante la exploración, se aplican técnicas de sincronización para garantizar un acceso controlado al laberinto por parte de los hilos y evitar conflictos en la información, así como una correcta verificación de las restricciones para llevar a cabo el correcto recorrido del laberinto y así evitar bucles o estados imposibles en la exploración.

En las siguientes secciones, se detallan aspectos relevantes del proyecto como la estrategia de solución empleada, el análisis de resultados, las lecciones aprendidas y los casos de pruebas realizados. Este documento funciona como registro y guía para comprender el proceso de desarrollo y los resultados alcanzados en la creación de la aplicación.

## Estrategia de solución

La estrategia de solución se basó en los siguientes pasos:

- **Lectura del Laberinto:** Se leerá el laberinto desde un archivo de texto, identificando correctamente las paredes y la posición de salida, almacenando los datos en una matriz para su futuro uso.
- **Control de Direcciones y Movimientos:** Cada hilo tendrá asignada una dirección (Arriba, Abajo, Izquierda, Derecha), y este se debe de mover en la misma dirección que fue creado hasta encontrar un obstáculo, la meta o un camino ya explorado en la misma dirección .
- **Creación de Hilos:** Se generará un hilo inicial que comenzará a explorar el laberinto en las coordenadas (0,0), cuando este encuentre bifurcaciones, creará nuevos hilos para explorar los caminos alternativos y seguirá haciendo esto por cada avance que haga hacia su dirección establecida.
- **Finalización de Hilos:** Los hilos finalizarán cuando lleguen a la salida, se encuentren con una pared o al detectar un camino previamente recorrido, de esta manera se eliminan los bucles infinitos o una posición imposible en el laberinto.
- **Visualización en consola:** Se mostrará en consola el progreso de cada hilo utilizando caracteres para representar el laberinto y los movimientos de los hilos manteniendo el "\*" y "/" como pared y meta respectivamente, adicional a esto se piensa añadir los caracteres de "<", ">", "v" y "^" para indicar las direcciones que llevan los hilos.

# Análisis de resultados

El análisis de los resultados revela que todas las áreas del proyecto se lograron con rendimiento del 100%, lo que refleja el éxito de la implementación. A continuación se presentan algunas notas adicionales para profundizar en cada aspecto del trabajo que se realizó.

Tarea	Porcentaje	Notas
Lectura del laberinto	100%	La lectura del laberinto se realizó correctamente a partir del archivo de texto proporcionado, se obtuvieron las dimensiones del laberinto y se almacenaron correctamente en la estructura de datos.
Creación de hilos	100%	Los hilos se crearon de la manera esperada para explorar el laberinto, se logró la generación de nuevos hilos en cada bifurcación y se manejaron correctamente las condiciones para evitar la creación de hilos en direcciones inválidas.
Direcciones y movimientos	100%	El control de direcciones y movimientos de los hilos se implementó con éxito, se logra verificar las posiciones visitadas por cada hilo y nos aseguramos que no se repitieran las direcciones en las mismas posiciones.
Finalización de hilos	100%	Los hilos finalizan de manera correcta al alcanzar la salida, topar con una pared o encontrar un camino previamente recorrido en la misma dirección, adicional a eso se logra desplegar la información correspondiente al finalizar cada hilo.
Visualización en consola	100%	Se concluyó de manera correcta la visualización en consola del progreso de los hilos, el laberinto y toda información relacionada, se usa un carácter diferente para representar cada dirección de movimiento de los hilos, lo que facilita la visualización del recorrido.

## Lecciones aprendidas

- **Manejo de hilos en C:** Una de las principales lecciones fue el cómo trabajar con hilos en C utilizando la biblioteca pthread, esto va desde la creación, gestión y sincronización de múltiples hilos para realizar tareas concurrentes de manera eficiente.
- **Sincronización con mutex:** Se aprendió sobre la importancia de la sincronización entre los hilos para evitar condiciones de carrera y garantizar el correcto manejo de los datos compartidos, el uso de mutex para controlar el acceso en partes del código fue de mucha ayuda para este proyecto.
- **Estructuras:** Logramos reforzar conceptos de las estructuras de datos complejas y su interacción con funciones y punteros, ya que teníamos tiempo de no relacionarnos con este tema, gracias a este proyecto logramos obtener una mayor comprensión y profundización en este tema.
- **Mejor entendimiento de la concurrencia y paralelismo:** Nos sumergimos en los conceptos de concurrencia y paralelismo al trabajar con múltiples hilos, con el proyecto logramos saber más sobre los desafíos de la ejecución de tareas concurrentes en la práctica y cómo diseñar programas aprovechando estos conceptos.

- **La importancia de las pruebas de software:** La importancia de las pruebas rigurosas fue evidente, ya que nos permitieron descubrir aspectos inesperados que no habíamos considerado inicialmente, a pesar de haber implementado el código con cuidado, nos encontramos con casos que no habíamos previsto, como la necesidad de validar si un espacio lateral al recorrido de un hilo era la meta, esto demuestra la importancia de realizar pruebas exhaustivas para abarcar la mayor cantidad de escenarios posibles y garantizar el correcto funcionamiento y la fiabilidad del software.

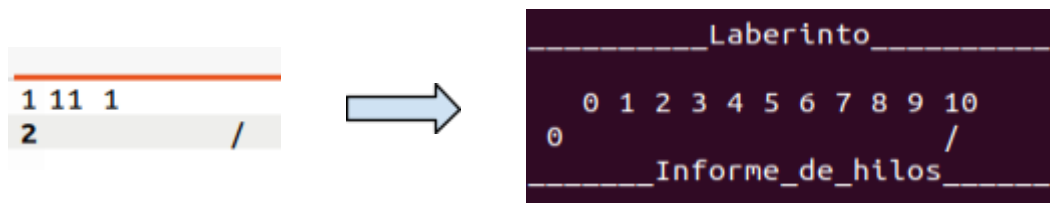
## Casos de pruebas

### 1. Laberinto simple sin obstáculos:

**Descripción:** Laberinto sin obstáculos ni paredes de 1x11, es decir un solo camino sin bifurcaciones hacia ningún lado.

**Resultados esperados:** El programa debe encontrar una ruta directa desde el punto de partida hasta la meta sin dar ningún error luego de pasar 10 espacios.

**Laberinto proporcionado:**



**Resultados obtenidos:** El programa genera y encuentra la ruta directa correctamente y muestra el mensaje de que se ha alcanzado la meta después de 10 movimientos.

```

Laberinto
0 1 2 3 4 5 6 7 8 9 10
0 > > > > > > /
Informe_de_hilos

Laberinto
0 1 2 3 4 5 6 7 8 9 10
0 > > > > > > /
Informe_de_hilos

Laberinto
0 1 2 3 4 5 6 7 8 9 10
0 > > > > > > /
Informe_de_hilos

Laberinto
0 1 2 3 4 5 6 7 8 9 10
0 > > > > > > /
Informe_de_hilos
->Hilo 0 ha alcanzado la meta en la posición (10,0) después de recorrer 10 espacios.<-

Hay un solo camino que conduce a la salida

```

## 2. Laberinto con un solo camino hacia la salida:

**Descripción:** Laberinto con un solo camino posible desde el punto de partida hasta la meta con algunas bifurcaciones.

**Resultados esperados:** El programa debe seguir el único camino posible y llegar a la meta sin problemas luego de 21 movimientos y generar 6 hilos.

**Laberinto proporcionado:**

```
1 11 11
2  *****
3  *           **
4  ***** **
5  ***** **
6  ***** **
7  ***** **
8  **** **
9  **** *****
10 **** *****
11 ****/******
12 *****
```



```
Laberinto
0 1 2 3 4 5 6 7 8 9 10
0  * * * * * * * *
1  *
2  * * * * * * * *
3  * * * * * * * *
4  * * * * * * * *
5  * * * * * * * *
6  * * * * * * * *
7  * * * * * * * *
8  * * * * * * * *
9  * * * * / * * * *
10 * * * * * * * *
Informe_de_hilos
```

**Resultados obtenidos:** El programa sigue el camino correctamente y llega a la meta sin problemas luego de recorrer 21 espacios y generar 6 hilos (toma en cuenta el índice 0).

```
Laberinto
0 1 2 3 4 5 6 7 8 9 10
0  > > * * * * * * *
1  * v > > > > > *
2  * * * * * * * v *
3  * * * * * * * v *
4  * * * * * * * v *
5  * * * * * * * v *
6  * * * * < < < v *
7  * * * * v * * * *
8  * * * * v * * * *
9  * * * * / * * * *
10 * * * * * * * *
Informe_de_hilos
->Hilo 5 ha alcanzado la meta en la posicion (4,9) después de recorrer 21 espacios.<-
Hay un solo camino que conduce a la salida
```

## 3. Laberinto con múltiples caminos hacia la salida:

**Descripción:** Laberinto con múltiples caminos posibles desde el punto de partida hasta la meta.

**Resultados esperados:** Debe detectar 2 caminos diferentes hacia la salida y dar información de los 2 hilos cuando estos lleguen a la salida.

**Laberinto proporcionado:**

```
1 11 11
2  *****
3  *           **
4  * ***** **
5  * ***** **
6  * ***** **
7  * ***** **
8  * ** **
9  * ** *****
10 * ** *****
11 * /******
12 *****
```



```
Laberinto
0 1 2 3 4 5 6 7 8 9 10
0  * * * * * * * *
1  *
2  * * * * * * * *
3  * * * * * * * *
4  * * * * * * * *
5  * * * * * * * *
6  * * * * * * * *
7  * * * * * * * *
8  * * * * * * * *
9  * * * * / * * * *
10 * * * * * * * *
Informe_de_hilos
```

**Resultados obtenidos:** Muestra ambas de forma correcta y genera la información de ambas por separado

```

Laberinto
 0 1 2 3 4 5 6 7 8 9 10
0 > * * * * *
1 * v > > > > *
2 * v * * * * v *
3 * v * * * * v *
4 * v * * * * v *
5 * v * * * * *
6 * v * * * *
7 * v * * * *
8 * v * * * *
9 * v > / * * *
10 * * * * *
Informe_de_hilos
->Hilo 4 ha alcanzado la meta en la posicion (4,9) después de recorrer 13 espacios.<-

```

```

Laberinto
 0 1 2 3 4 5 6 7 8 9 10
0 > * * * * *
1 * v > > > > *
2 * v * * * * v *
3 * v * * * * v *
4 * v * * * * v *
5 * v * * * * v *
6 * v * * < < < v *
7 * v * * v * * *
8 * v * * v * * *
9 * v > / * * *
10 * * * * *
Informe_de_hilos
->Hilo 6 ha alcanzado la meta en la posicion (4,9) después de recorrer 21 espacios.<-

Hay 2 caminos diferentes que conducen a la salida

```

#### 4. Laberinto sin salida:

**Descripción:** Laberinto en el que no hay una meta disponible.

**Resultados esperados:** Debe detectar todos los caminos posibles y generar un informe de que no se encontró ruta hacia la salida.

**Laberinto proporcionado:**

```

1 11 11
2  * * * * *
3 *          **
4 * * * * * **
5 * * * * * **
6 * * * * * **
7 * * * * * **
8 * **      **
9 * ** * * *
10 * ** * * *
11 *  * * * *
12 * * * * *

```



```

Laberinto
 0 1 2 3 4 5 6 7 8 9 10
0 * * * * *
1 *          **
2 * * * * * **
3 * * * * * **
4 * * * * * **
5 * * * * * **
6 * **      **
7 * ** * * *
8 * ** * * *
9 *  * * * *
10 * * * * *
Informe_de_hilos

```

**Resultados obtenidos:** Se logra generar la exploración de todo el laberinto e informa de que no hay ruta que dirija a la salida.

```
-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0 > > * * * * * * * *
1 * v > > > > > * *
2 * v * * * * * v * *
3 * v * * * * * v * *
4 * v * * * * * v * *
5 * v * * * * * v * *
6 * v * * < < < < v * *
7 * v * * v * * * * *
8 * v * * v * * * * *
9 * v > > * * * * * *
10 * * * * * * * * *
-----Informe_de_hilos-----
->Hilo 6 ha alcanzado una pared posicion (4,9) después de recorrer 20 espacios.<-

No se encontro una salida del laberinto!!! :0
```


## 5. Laberinto completo:

**Descripción:** Laberinto con diferentes caminos y 2 metas con varios caminos posibles.

**Resultados esperados:** El programa debe manejar eficientemente el laberinto y encontrar una ruta hacia la salida siguiendo todas las restricciones necesarias.

**Laberinto proporcionado:**

```
1 11 11
2  **** * **
3  *      *
4  * ** * * **
5  *  * *** **
6  * **  *  *
7  * **** * **
8  *  *    **
9  * ** *** /
10 * ** *** **
11          /**
12 *****
```



```
-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0  * * * * * * * *
1  *      * * *
2  *  *  *  * * *
3  *  *  * * * *
4  *  *  *  * *
5  *  * * * * * *
6  *  *  *  * *
7  *  *  * * * /
8  *  *  * * * *
9  *  *  *  * / *
10 *  *  * * * *
-----Informe_de_hilos-----
```

**Resultados obtenidos:** Maneja correctamente el laberinto, encuentra todas las rutas posibles y verifica todas las restricciones de manera correcta.

```
-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0  < ^ * * * * ^ * ^ *
1  * < < < < < < ^ > *
2  * ^ * * v * v * ^ * *
3  * ^ > * v * * * ^ * *
4  * ^ * * v > > * ^ > *
5  * ^ * * * * * * ^ * *
6  * ^ > * ^ > > > * *
7  * ^ * * v * * * v > /
8  * ^ * * v * * * v * *
9  < < < < v > > / * *
10 * * * * * * * * *
-----Informe_de_hilos-----
->Hilo 22 ha alcanzado una pared posicion (0,1) después de recorrer 32 espacios.<-

Hay 3 caminos diferentes que conducen a la salida
```



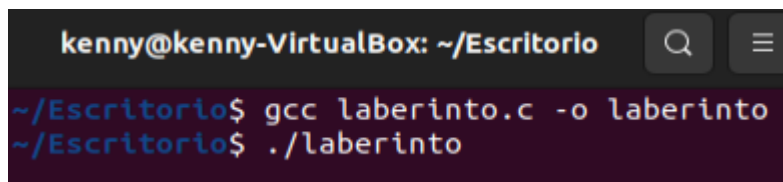
# Comparación con creación de Procesos Fork

Aspecto	Procesos con fork	Hilos (Threads)
Espacio de memoria	Con la creación de procesos utilizando fork, cada proceso tendría su propio espacio de memoria independiente, esto significa que cada explorador del laberinto estaría completamente aislado de los demás, lo que proporcionaría una mayor seguridad y estabilidad en caso de fallos.	Al utilizar hilos (threads), todos los hilos comparten el mismo espacio de memoria, lo que simplifica la comunicación y sincronización entre los exploradores del laberinto, sin embargo, esto también significa que un fallo en un hilo puede afectar a todos los demás hilos en el mismo proceso.
Comunicación y sincronización	La comunicación entre procesos con fork requeriría el uso de mecanismos como tuberías (pipes), señales o memoria compartida, lo que agrega complejidad al proyecto y puede requerir más código para manejar la transferencia de datos entre los procesos.	La comunicación entre hilos es más directa, ya que pueden acceder y modificar las mismas variables globales y estructuras de datos, lo que lo hace más fácil.
Rendimiento	La creación y terminación de procesos con fork pueden ser más lentas en comparación con los hilos, esto podría resultar en un tiempo de ejecución más largo, especialmente si se necesita estar creando y terminando muchos procesos.	La creación y terminación de hilos son generalmente más rápidas, lo que puede resultar en un tiempo de ejecución más corto y una exploración más eficiente del laberinto.
Estabilidad y seguridad	Cada proceso con fork estaría completamente aislado, lo que significa que los movimientos serían independientes y no necesitarían ser coordinados con otros procesos.	En el caso de los hilos, los movimientos deben coordinarse para evitar colisiones y asegurar que cada hilo cubra áreas únicas del laberinto, esto requiere sincronización entre los hilos utilizando mecanismos como mutex o semáforos.

# Manual de usuario

## 1. Compilación del proyecto:

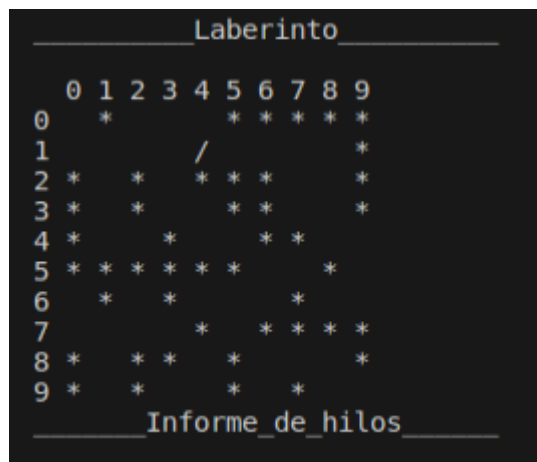
- Abrir una terminal en el directorio donde se encuentra el código fuente del proyecto y el txt con un formato de laberinto compatible.
- Ejecutar el siguiente comando para compilar el código fuente y generar el ejecutable:
  - **`gcc laberinto.c -o laberinto -pthread`**
- Una vez compilado, se ejecuta el programa utilizando el siguiente comando en la terminal:
  - **`./laberinto`**



```
kenny@kenny-VirtualBox: ~/Escritorio
~/Escritorio$ gcc laberinto.c -o laberinto
~/Escritorio$ ./laberinto
```

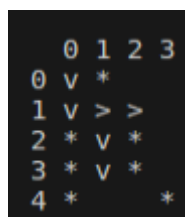
## 2. Ejecución del programa:

- Al ejecutar el programa, se inicia la exploración del laberinto.



```
Laberinto
 0 1 2 3 4 5 6 7 8 9
0  *      * * * * *
1      /          *
2 *  *  * * *      *
3 *  *      * *      *
4 *      *  * *      *
5 * * * * * *      *
6  *  *      *      *
7      *  * * * *
8 *  * *  *      *
9 *  *      *      *
Informe_de_hilos
```

- Cada hilo se representa con un carácter en el laberinto indicando su dirección y posición mediante la consola.



```
 0 1 2 3
0 v *
1 v > >
2 * v *
3 * v *
4 *      *
```

- El programa continuará ejecutándose hasta que todos los hilos hayan alcanzado la salida del laberinto o hayan llegado a una pared que les impida avanzar.

- Una vez que la exploración haya terminado, el programa mostrará un mensaje indicando cuántos caminos diferentes conducen a la salida del laberinto y la cantidad de movimientos que les tomó en llegar.

```

-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0 ^ * < < > * * * * *
1 < < v < < < ^ ^ *
2 * v * v * * * ^ ^ *
3 * v * v > * * < > *
4 * v > * v > * * v > *
5 * * v * * * * * v *
6 ^ * v * * * * < v *
7 < < v > * * * * *
8 * v * * * * * *
9 * v * * * * *
10 * / * * * * * * *
-----Informe_de_hilos-----
->Hilo 40 ha alcanzado una pared posicion (7,6) después de recorrer 16 espacios.<-

Hay un solo camino que conduce a la salida
kenny@kenny-VirtualBox:~/Escritorio$ s

```

**Nota:** Si se desea modificar o cambiar el laberinto se debe de hacer en el documento llamado laberinto.txt, guardar los cambios y repetir el proceso anterior, es necesario asegurarse de no excluir los espacios en blanco al final de cada línea ya que estos también son tomados en cuenta como caracteres y de no ir pueden afectar el formato del laberinto, aunque el error llega a ser meramente visual, como se puede observar en la imagen de la izquierda, el laberinto no se crea de manera adecuada y genera cierta perturbación, mientras que el de la derecha lo hace de manera correcta.

```

-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0 > * ^ ^ ^ * * * * *
1 v > > v > *
2 * v * v * * * *
3 * v * * * * *
4 * v * * * * *
5 * * * * * *
6 * * * * *
7 * * * * *
8 * * * * *
9 * * * * *
10 * / * * * * *

```

```

-----Laberinto-----
  0 1 2 3 4 5 6 7 8 9 10
0 > * < > > * * * * *
1 v > > v > > *
2 * v * v * * * *
3 * v * v > * * *
4 * v * * * * *
5 * * * * * *
6 * * * * *
7 * * * * *
8 * * * * *
9 * * * * *
10 * / * * * * *

```

# Bitácora de trabajo

## **Del 25 al 26 marzo:**

- Revisión y comprensión de los requisitos del proyecto.
- Investigación sobre el manejo de hilos en C.
- Discusión y planificación de la estrategia de implementación.

## **Del 26 marzo al 01 abril:**

- Implementación de la lectura del laberinto desde un archivo de texto.
- Desarrollo de la creación de hilos y control de direcciones.
- Solución de problemas relacionados con la sincronización de hilos.

## **Del 02 al 04 abril:**

- Depuración y pruebas exhaustivas del programa.
- Resolución de errores encontrados durante las pruebas.
- Documentación del código y redacción de la documentación del proyecto.

# Bibliografía

Martinez, S.(11 de julio de 2018). *Cómo funciona la función fork()*. Stackoverflow.  
Recuperado de  
<https://es.stackoverflow.com/questions/179414/como-funciona-la-funci%C3%B3n-fork>

Early Adopters, Universidad de Murcia. (6 de febrero de 2012 ). *Procesos e Hilos en C*.  
Recuperado de  
[https://www.um.es/earlyadopters/actividades/a3/PCD\\_Activity3\\_Session1.pdf](https://www.um.es/earlyadopters/actividades/a3/PCD_Activity3_Session1.pdf)

ProgramaTutos.(11 de octubre de 2023). *Aprende a sincronizar hilos en C con Mutex*  
[Video]. YouTube.  
[https://www.youtube.com/watch?v=q66qT0f60Ko&ab\\_channel=ProgramaTutos](https://www.youtube.com/watch?v=q66qT0f60Ko&ab_channel=ProgramaTutos)

Makigas.(28 de junio de 2015). *Tutorial de C – 8. Estructuras* [Video]. YouTube.  
[https://www.youtube.com/watch?v=leHnsH0gx2A&t=588s&ab\\_channel=makigas](https://www.youtube.com/watch?v=leHnsH0gx2A&t=588s&ab_channel=makigas)

Santigamo.(19 de abril de 2017). *¿Cómo leer .txt en Lenguaje C?*. Stackoverflow.  
Recuperado de  
<https://es.stackoverflow.com/questions/64016/c%C3%B3mo-leer-txt-en-lenguaje-c>

Jinku, H.(12 de octubre de 2023). *Obtener el ID del hilo en C*. DelftStack. Recuperado de  
<https://www.delftstack.com/es/howto/c/pthread-get-thread-id-in-c/>