

# **PROYECTO DE UNIDAD I**

# **MINISHELL EN C++**

# **PARA LINUX**

## **INTEGRANTES:**

Lizeth Hancco Calizaya  
Néstor André Alarcón Luque

2023-119022  
2023-119002

# Introducción

El presente proyecto consiste en la implementación de una mini-shell en C++ para sistemas Linux, que permite ejecutar comandos del sistema con soporte para procesos hijo, redirecciones y control de errores. La implementación pone en práctica llamadas POSIX clave como fork, execvp, waitpid y dup2, integrando además buenas prácticas de modularidad y pruebas en múltiples distribuciones.

# Objetivo

Implementar un intérprete de comandos tipo mini-shell en C++ sobre sistemas Linux, capaz de ejecutar comandos del sistema, manejar procesos y realizar redireccionamiento salida y manejo de errores y rutas de comandos.

# Alcance

la implementación de las características base de una mini-shell: un prompt personalizado, la resolución de rutas tanto absolutas como relativas, la ejecución de comandos mediante procesos utilizando fork() y exec(), el manejo de errores utilizando perror(), la redirección de la salida estándar (>) y el comando interno 'salir' para terminar la ejecución de la mini-shell.



# Arquitectura y diseño

SRC

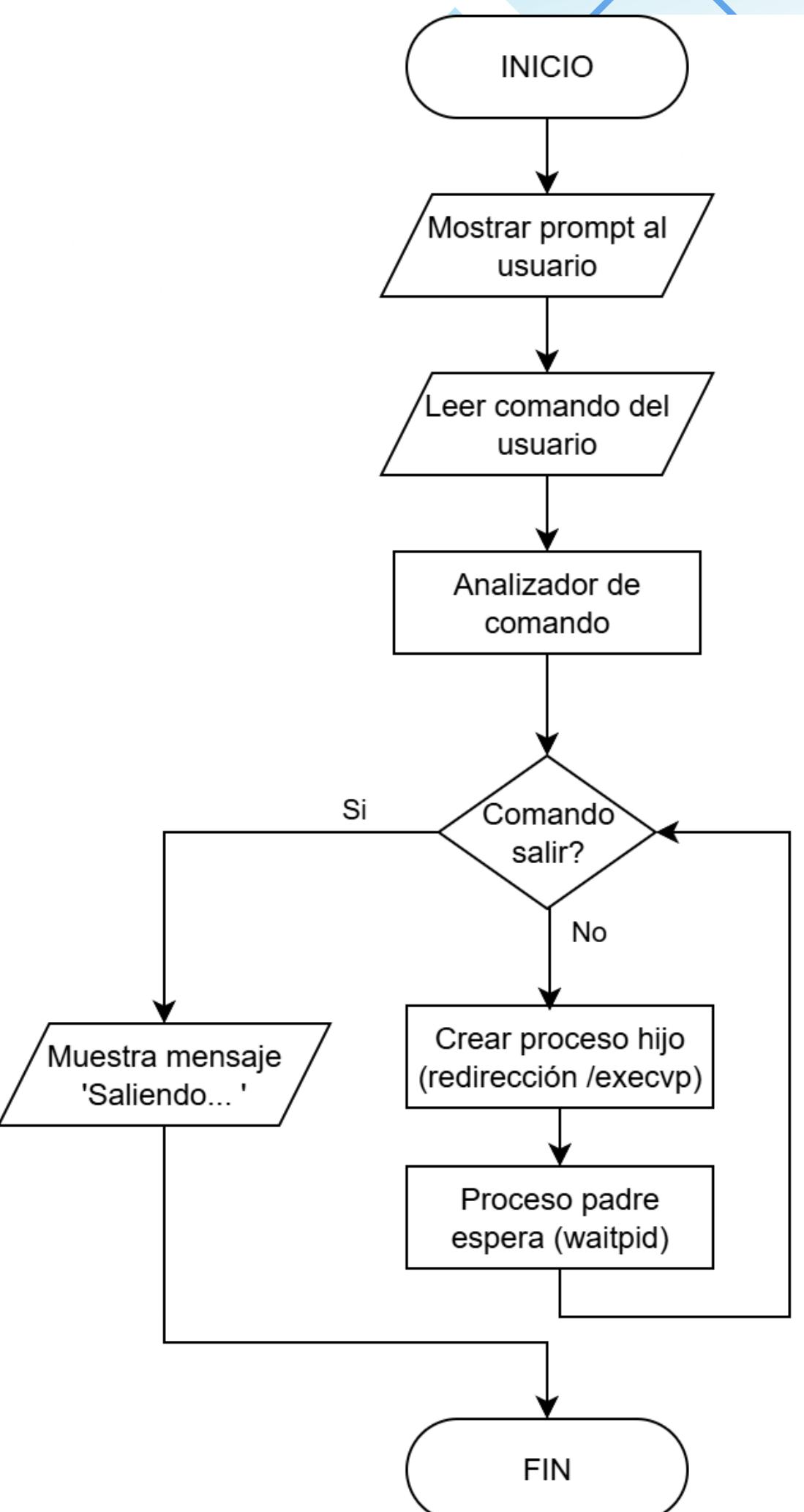
Donde se incluirá los archivos de código fuente como el main.cpp, analizador.cpp y ejecutor.cpp  
include:

INCLUDE

Donde se incluirá los archivos como el analizador.hpp y ejecutor.hpp

DOCS

Donde se incluirá el informe técnico en pdf, diapositivas en pdf y una carpeta de imágenes donde estarán las capturas de las pruebas realizadas del minishell.



# Detalles de implementación

- fork(), crea un proceso hijo.
- execvp, ejecuta el comando solicitado.
- waitpid, el proceso padre espera.
- access(), verificación de existencia y permisos.
- open(), crear y abrir el archivo para la redirección.
- dup2(), redirigir la salida estándar al archivo.
- close(), cerrar el descriptor de archivo.
- perror(), imprimir la descripción del error
- pipe(), para conectar la salida de un proceso con la entrada de otro

# Pruebas y resultados

```
Minishell$ ls  
analizador.cpp ejecutor.cpp main.cpp minishell  
Minishell$ █
```

```
Minishell$ /usr/bin/ls  
analizador.cpp ejecutor.cpp main.cpp minishell  
Minishell$ █
```

```
Minishell$ comando_invalido  
Error al ejecutar el comando : No such file or directory  
Minishell$ █
```

```
Minishell$ salir  
Saliendo de la miniShell...  
unjbg@unjbg-VirtualBox:~/Descargas/S0-Proyecto/src$ █
```

```
Minishell$ ls > salida.txt  
Minishell$ cat salida.txt  
analizador.cpp  
ejecutor.cpp  
main.cpp  
minishell  
salida.txt  
Minishell$ █
```

```
Minishell$ ls >  
Error: falta el nombre del archivo después de '>'  
Minishell$ █
```

# Pruebas y resultados

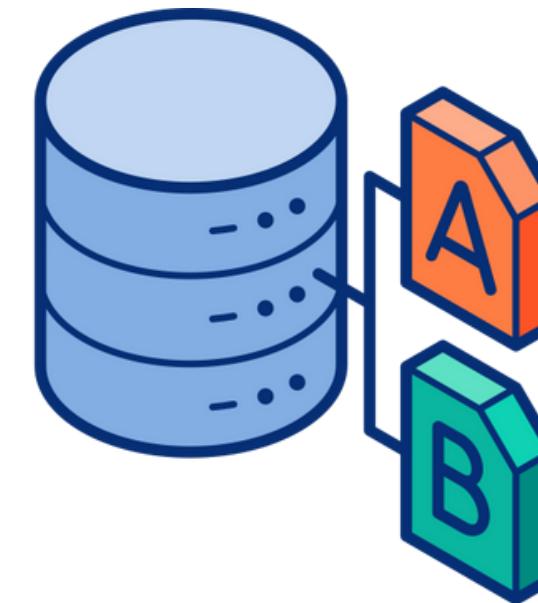
```
MiniShell$ ls -l | grep .txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005      39 Oct 15 06:16 datos.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005      70 Oct 15 00:30 listado.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005     10 Oct 15 06:17 resultado.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005     63 Oct 14 19:06 salida.txt
MiniShell$
```

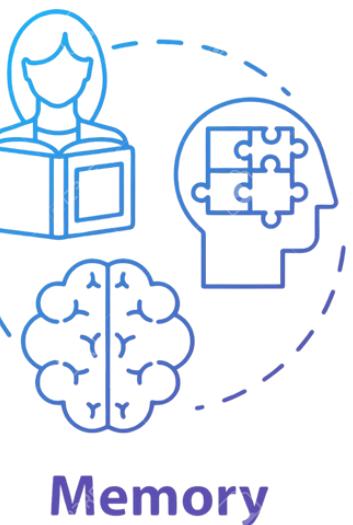
```
MiniShell$ sleep 10 &
Proceso en segundo plano: 108743
MiniShell$ ls -l
total 76
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005    315 Oct 14 19:06 analizador.cpp
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005    39 Oct 15 06:16 datos.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005   8149 Oct 15 06:23 ejecutor.cpp
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005    70 Oct 15 00:30 listado.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005   603 Oct 14 19:06 main.cpp
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005  67416 Oct 15 06:14 minishell
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005    10 Oct 15 06:17 resultado.txt
-rwxrwxrwx 1 nestolarcon2005 nestolarcon2005    63 Oct 14 19:06 salida.txt
MiniShell$ |
```

```
MiniShell$ echo manzana > datos.txt
MiniShell$ echo banana >> datos.txt
MiniShell$ echo manzana y naranja >> datos.txt
MiniShell$ grep manzana < datos.txt
manzana
manzana y naranja
MiniShell$
```

# Concurrencia y sincronización

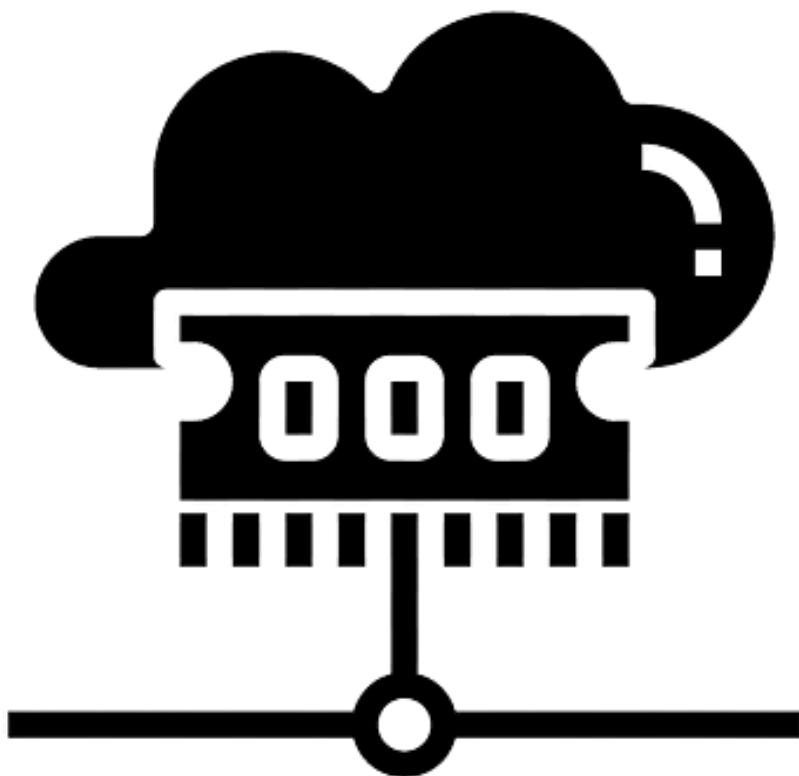
Desde la implementación de las extensiones se empezó a usar procesos en segundo plano, pero no una paralelización que requiera de funciones como `pthread_create()`





# Gestión de memoria

Para la memoria se utilizó estructuras estándar de C++ como los vectores, ya que no se sabe el tamaño para almacenar , pues el usuario es quien digita el comando.

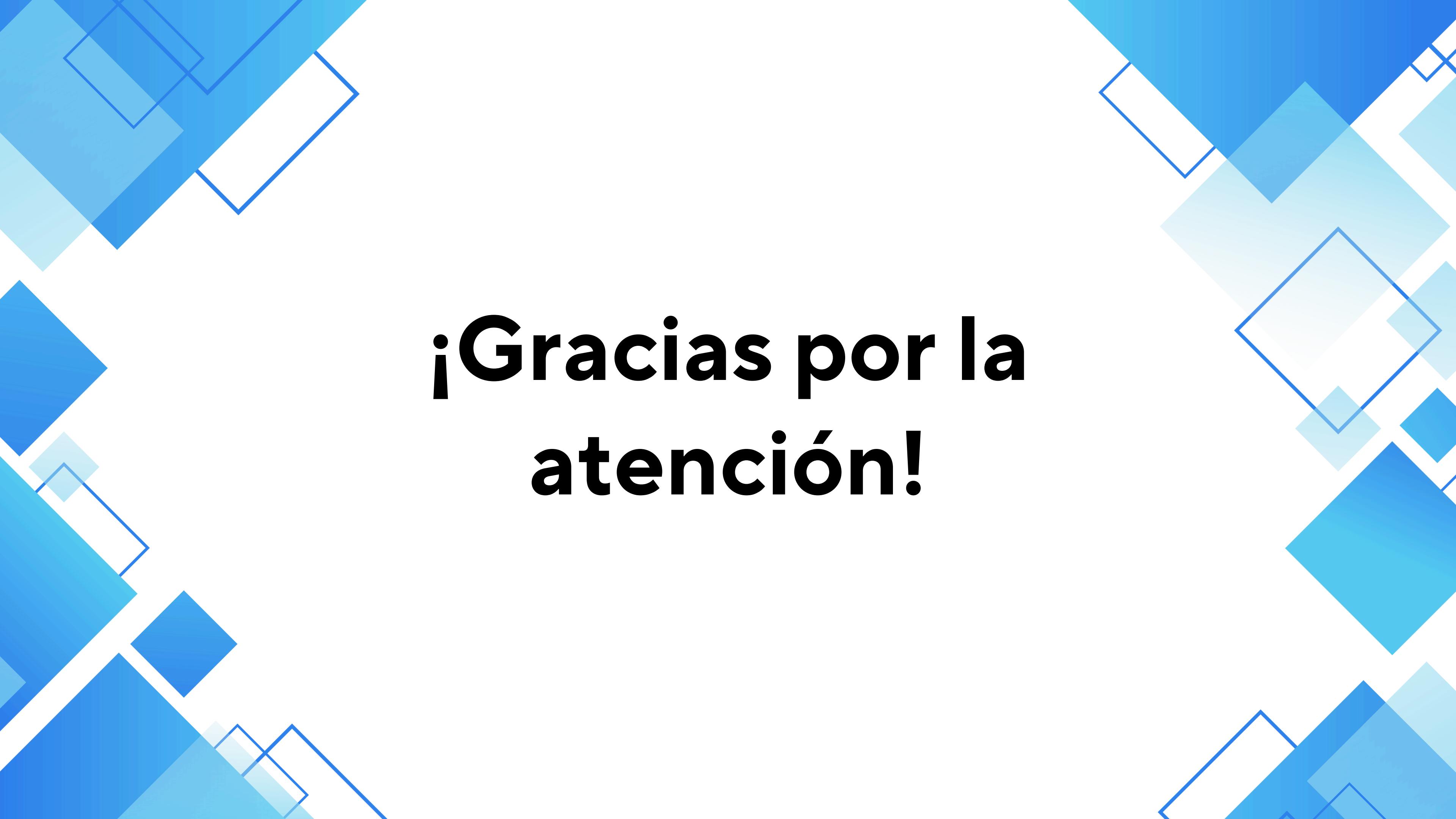


# Conclusiones



La minishell se logró implementar correctamente con las funciones base de prompt, ejecución de comandos, la redirección de salida y el manejo de errores. Donde se utilizó APIs POSIX para demostrar la comprensión de la creación y control de procesos.

Se utilizó un diseño del software empleando cabeceras y varios archivos, así como herramientas como git, lo que evidencia la comprensión de estos conceptos importantes del desarrollo de software profesional.

The background features a dynamic arrangement of overlapping blue rectangles of varying shades (light blue, medium blue, dark blue) and thin blue outlines, creating a sense of depth and motion.

**¡Gracias por la  
atención!**