

Reporte técnico: Proyecto final de Sistemas Operativos y Laboratorio

1. Información del proyecto

- **Título del Proyecto:** [Initial reverse]
- **Curso/Materia:** Sistemas Operativos
- **Integrantes:** [Johana Liseth Sevillano Herrera (johana.sevillano@udea.edu.co), Angi Sirley Hoyos Ruíz (asirley.hoyos@udea.edu.co), Angie Paola Yarce Gómez (angie.yarceg@udea.edu.co)]
- **Fecha de Entrega:** [Julio 10 de 2025]

2. Introducción

2.1. Objetivo del proyecto

El objetivo principal del proyecto "Initial Reverse" es desarrollar una aplicación en lenguaje C que invierta el orden de las líneas de un archivo de texto. Esta aplicación busca resolver la necesidad de manipular el contenido de archivos de texto, específicamente invirtiendo su orden, y demostrar la capacidad de manejar operaciones de entrada/salida de archivos, gestión dinámica de memoria y, opcionalmente, la implementación de una interfaz gráfica de usuario (GUI) con GTK para facilitar su uso. El proyecto también se propone demostrar un manejo robusto de errores y validaciones, como la verificación de archivos vacíos o inexistentes.

2.2. Motivación y justificación

La elección de este proyecto surge de la necesidad de comprender y aplicar conceptos fundamentales de los sistemas operativos, como el manejo de archivos, la asignación de memoria dinámica y la interacción con el sistema a través de llamadas a funciones de biblioteca estándar de C. La inversión de líneas en un archivo, aunque aparentemente simple, engloba varios desafíos técnicos que permiten explorar en profundidad estos conceptos.

La relevancia en el contexto de los sistemas operativos radica en que las operaciones de archivo son una parte integral de cualquier sistema. Este proyecto permite entender cómo los programas interactúan con el sistema de archivos, cómo se gestiona la memoria de manera eficiente para procesar datos de tamaño variable (líneas de un

archivo), y cómo se pueden construir herramientas útiles para la manipulación de texto. Además, la inclusión de una interfaz gráfica de usuario con GTK añade una capa de interacción con el usuario, lo cual es fundamental en el desarrollo de aplicaciones modernas y demuestra la capacidad de integrar componentes de bajo nivel con bibliotecas de GUI. Es interesante e importante porque permite abordar desde una perspectiva práctica la manipulación de datos textuales, un problema común en diversos ámbitos de la computación, y muestra la versatilidad del lenguaje C para desarrollar tanto herramientas de consola como aplicaciones con interfaz gráfica. Es interesante e importante porque permite abordar desde una perspectiva práctica la manipulación de datos textuales, un problema común en diversos ámbitos de la computación, y muestra la versatilidad del lenguaje C para desarrollar tanto herramientas de consola como aplicaciones con interfaz gráfica.

3. Marco teórico / Conceptos fundamentales

El proyecto "Initial Reverse" se basa en los siguientes conceptos fundamentales y tecnologías:

Lenguaje de programación C: C es el lenguaje principal utilizado debido a su eficiencia, control de bajo nivel sobre la memoria y acceso directo a las llamadas al sistema operativo. Esto permite una implementación optimizada de las operaciones de archivo y gestión de memoria.

Manejo de archivos (fopen(), fclose(), fprintf()): Estas funciones son esenciales para la interacción con el sistema de archivos. fopen() se utiliza para abrir archivos en diferentes modos (lectura "r", escritura "w"), fclose() para cerrarlos, liberando los recursos del sistema, y fprintf() para escribir datos en el archivo de salida o en la consola.

Gestión dinámica de memoria (malloc(), realloc(), free()): Dado que el tamaño de las líneas y el número total de líneas en un archivo son desconocidos en tiempo de compilación, el proyecto utiliza memoria dinámica.

malloc(): Se emplea para asignar bloques de memoria en el heap. En este proyecto, se usa inicialmente para el arreglo de punteros a líneas y para el buffer de getline().

realloc(): Permite redimensionar un bloque de memoria previamente asignado. Es crucial para el arreglo dinámico de punteros a líneas (char **lines), que se expande a medida que se leen más líneas del archivo de entrada.

free(): Se usa para liberar la memoria asignada dinámicamente, previniendo fugas de memoria. Es fundamental liberar tanto las copias de las líneas como el arreglo de punteros a líneas al finalizar el programa.

El informe de Valgrind confirma que "Se liberaron todos los bloques del montón--no es posible que haya fugas", lo que indica una correcta gestión de la memoria.

Lectura dinámica de líneas (getline()): Esta función es vital para leer líneas de un archivo sin conocer su longitud de antemano. `getline()` asigna o reasigna memoria según sea necesario para almacenar la línea leída, lo que la hace ideal para procesar archivos con líneas de longitud variable.

2.3. Alcance del proyecto:

Duplicación segura de cadenas (strdup()): Después de leer una línea con `getline()`, `strdup()` se utiliza para crear una copia separada de la cadena de caracteres. Esto es necesario porque `getline()` reutiliza su buffer, y si no se duplica la línea, las líneas anteriores se sobrescribirán.

Manejo de strings (strcmp(), strcat(), strlen()):

strcmp(): Se utiliza para comparar cadenas de caracteres, por ejemplo, para verificar los argumentos de la línea de comandos como `-o`.

strcat(): Concatena cadenas, utilizado en la versión GUI para construir el resultado invertido antes de mostrarlo.

strlen(): Calcula la longitud de una cadena, esencial para determinar el tamaño total de la memoria necesaria para concatenar las líneas invertidas en la versión GUI.

GTK 3: Es una biblioteca de código abierto utilizada para crear interfaces gráficas de usuario. En este proyecto se utiliza GTK para la interfaz gráfica, incluyendo elementos como **GtkWindow**, **GtkButton**, **GtkTextView** y **GtkFileChooserDialog**.

Manejo de errores y validaciones: El proyecto incluye validaciones robustas para mejorar la experiencia del usuario y la fiabilidad:

- Verificación de existencia de archivo (`!file`).
- Detección de archivos vacíos (`is_file_empty()`).
- Mensajes de error y advertencia descriptivos para el usuario, tanto en consola (con colores) como en la GUI (diálogos emergentes).

4. Diseño e implementación

4.1. Diseño de la solución

El diseño de la solución de "Initial Reverse" se compone de dos módulos principales: una versión de consola y una versión con interfaz gráfica (GUI), compartiendo la lógica central de inversión de líneas.

El primer diseño es del modulo de consola:

El módulo de consola sigue un flujo secuencial:

Validación de argumentos: Al iniciar, el programa verifica que se haya proporcionado al menos un archivo de entrada. Opcionalmente, se puede especificar un archivo de salida usando la bandera -o.

Apertura del archivo de Entrada: Se intenta abrir el archivo especificado en modo lectura. Si falla, se muestra un mensaje de error y el programa termina.

Verificación de archivo vacío: Se comprueba si el archivo de entrada está vacío. Si lo está, se emite una advertencia y el programa finaliza.

Lectura de líneas: Se utiliza `getline()` en un bucle para leer cada línea del archivo. Las líneas leídas se duplican con `strdup()` y se almacenan en un arreglo dinámico de punteros a char (`char **lines`). Este arreglo se redimensiona con `realloc()` a medida que se necesitan más líneas.

Cierre del archivo de entrada: Una vez leídas todas las líneas, el archivo de entrada se cierra.

Determinación de salida: Se verifica si se especificó un archivo de salida. Si es así, se abre ese archivo en modo escritura; de lo contrario, la salida se redirige a la consola (`stdout`). Si la apertura del archivo de salida falla, se muestra un error.

Impresión invertida: Las líneas almacenadas en el arreglo se imprimen en el orden inverso (desde la última hasta la primera) en el destino de salida (archivo o consola).

Liberación de memoria: Finalmente, toda la memoria asignada dinámicamente (las líneas individuales y el arreglo de punteros) se libera usando la función auxiliar `free_lines()` para evitar fugas de memoria.

4.2. Tecnologías y herramientas

Para el desarrollo del proyecto Initial Reverse, se utilizaron las siguientes tecnologías y herramientas:

- Lenguaje de programación: C
- Biblioteca gráfica: GTK+ 3.0 para la interfaz gráfica (GUI)
- Entorno de desarrollo: Virtual Box
- Sistema operativo: Ubuntu
- Compilador: GCC (GNU Compiler Collection)
- Herramientas adicionales:
- pkg-config para enlazar GTK en la compilación.
- Terminal para ejecución de pruebas del programa en consola.

4.3. Detalles de implementación

El proyecto cuenta con dos formas principales de interacción:

- Una interfaz gráfica (archivo `gui.c`)
- Una versión en consola (archivo `main.c`)

Funcionalidades clave implementadas:

- **Lectura de archivos línea por línea** utilizando la función `getline()`.
- **Almacenamiento dinámico** de líneas en un arreglo de punteros con `malloc`, `realloc`, y `strdup`.
- **Inversión del orden** de las líneas mediante un ciclo inverso al momento de imprimir o concatenar.
- **Verificación de errores** como archivo vacío, fallo al abrir archivos, y control de memoria dinámica.
- En la versión GUI, se utilizan **cuadros de diálogo** para mostrar mensajes de advertencia o error usando `gtk_message_dialog_new`.

Estructuras de datos utilizadas:

- **Arreglo dinámico de cadenas** (`char **lines`) para almacenar las líneas leídas.
- **Cadenas dinámicas** para cada línea leída y para la concatenación del resultado en GUI.

Archivos de configuración:

No se requieren archivos de configuración específicos. Todo el comportamiento está controlado desde los argumentos de línea de comandos (en la versión consola) o mediante botones e interacciones en la GUI.

5. Pruebas y evaluación

5.1. Metodología de pruebas

Para validar la funcionalidad y estabilidad del proyecto Initial Reverse se utilizó una **metodología de pruebas funcionales**, incluyendo:

- **Pruebas manuales** con entrada real (`input.txt`).
- **Pruebas automáticas por lotes** con múltiples archivos de prueba simulando distintos escenarios.
- **Pruebas tanto en GUI como en modo consola.**

Se diseñaron **casos de prueba específicos** para evaluar:

- Archivos vacíos.
- Archivos con una sola línea.
- Archivos con múltiples líneas.
- Manejo de errores como archivos inexistentes.
- Archivos con caracteres especiales o líneas en blanco.

5.2. Casos de prueba y resultados

ID Caso de prueba	Descripción del caso de prueba	Resultado esperado	Resultado obtenido	Éxito/Fallo
CP-001	Archivo con múltiples líneas	Se invierten las líneas correctamente	Líneas invertidas mostradas correctamente	Éxito
CP-002	Archivo vacío	Mensaje de advertencia	“⚠ Advertencia: El archivo está vacío.”	Éxito
CP-003	Archivo inexistente	Error al abrir el archivo	“✖ Error: No se pudo abrir el archivo de entrada”	Éxito-002
CP-004	Archivo con una sola línea	Se imprime la misma línea	Línea única impresa sin cambios	Éxito
CP-005	Archivo con caracteres especiales y tides	Se mantienen caracteres especiales en reverso	Se respetan los caracteres especiales	Éxito
CP-006	Archivo con líneas en blanco entre líneas	Se mantienen líneas vacías al invertir	Se visualizan líneas en blanco correctamente	Éxito

5.3. Evaluación del rendimiento (si aplica)

ID Caso de prueba	Nro líneas	Tiempo Real	Tiempo de CPU user + sys	
CP-001	6	0m0,011s	0m0,08s	0m0,02s
CP-002	0	0m0,05s	0m0,04s	0m0,01s
CP-003	0	0m0,010s	0m0,08s	0m0,01s
CP-004	1	0m0,010s	0m0,09s	0m0,01s
CP-005	9	0m0,011s	0m0,08s	0m0,01s
CP-006	28	0m0,08s	0m0,07s	0m0,00s

Tabla de rendimiento

Métrica	Resultado
Tiempo de ejecución promedio	0.0287 segundos
Uso de memoria (Valgrind)	Sin fugas detectadas
Archivos probados	5
Escalabilidad	Adecuada para archivos pequeños

5.4. Problemas encontrados y soluciones

Durante el desarrollo de Initial Reverse, se presentaron diversos retos técnicos relacionados con la lectura de archivos, la gestión dinámica de memoria y la integración con bibliotecas externas (GTK). A continuación, se describen los principales problemas encontrados y las soluciones aplicadas:

1. No se detectaban archivos vacíos

Problema:

Cuando el archivo estaba vacío, el programa no daba ninguna advertencia y simplemente no mostraba salida, lo que causaba confusión.

Solución:

Se creó la función `is_file_empty()` para verificar si el archivo de entrada no contenía datos antes de iniciar la lectura. En caso afirmativo, se muestra una advertencia en consola o en la GUI.

2. Argumentos no reconocidos en consola

Problema:

Cuando el usuario usaba mal la bandera `-o` o ingresaba argumentos en orden incorrecto, el programa fallaba.

Solución:

Se mejoró la lógica de análisis de argumentos para validar correctamente la posición de `-o` y su argumento siguiente (archivo de salida), mostrando errores claros si faltaba alguno.

3. Dificultades con la integración de GTK

Problema:

Durante la compilación de la interfaz gráfica (`gui.c`), aparecían errores relacionados con bibliotecas no encontradas.

Causa:

No se estaban incluyendo correctamente los flags de GTK al compilar.

Solución:

Se usó el comando `pkg-config --cflags --libs gtk+-3.0` dentro del `Makefile`, asegurando la correcta vinculación con la biblioteca GTK. También se documentó la necesidad de tener GTK3 instalado en el sistema.

6. Conclusiones

El desarrollo del proyecto Initial Reverse permitió cumplir satisfactoriamente con los objetivos propuestos al inicio del curso. Se logró construir una herramienta funcional en lenguaje C capaz de leer un archivo de texto, invertir el orden de sus líneas y mostrar o guardar el resultado, utilizando para ello un manejo adecuado de operaciones de archivos, memoria dinámica y validaciones robustas.

Durante el proceso, se aplicaron de manera práctica múltiples conceptos clave del curso de Sistemas Operativos, tales como:

- El manejo de archivos mediante funciones como `fopen`, `getline` y `fprintf`.
- La gestión de memoria dinámica utilizando `malloc`, `realloc` y `free`, asegurando eficiencia y evitando fugas de memoria (verificado con Valgrind).
- El diseño de una interfaz gráfica (GUI) con GTK+, lo cual permitió explorar la interacción entre el sistema y el usuario a través de eventos, ventanas y diálogos de error.
- El uso de estructuras de datos dinámicas para almacenar y procesar información variable en tiempo de ejecución, como arreglos de cadenas.

Adicionalmente, se evaluó el rendimiento del programa con distintos archivos, comprobando que mantiene un tiempo de ejecución óptimo y un uso eficiente de la memoria, incluso con archivos más grandes. Los resultados de las pruebas confirman que la aplicación es estable, correcta y escalable para casos de uso comunes.

Más allá de los requerimientos iniciales, se diseñó el sistema de forma modular, lo que permitiría en el futuro incorporar mejoras, como invertir palabras dentro de cada línea, procesar múltiples archivos o exportar en otros formatos. Esto demuestra no solo un dominio de los aspectos técnicos, sino también una visión de escalabilidad y mantenibilidad del software.

En resumen, el proyecto fortaleció el entendimiento de cómo los programas interactúan con el sistema operativo, especialmente en relación con la gestión de recursos y la entrada/salida. Al mismo tiempo, evidenció la versatilidad del lenguaje C para el desarrollo tanto de aplicaciones de consola como con interfaz gráfica, consolidando habilidades fundamentales para el perfil profesional de los integrantes del equipo.

7. Trabajo futuro (Opcional)

A pesar de que el programa Initial Reverse cumple satisfactoriamente con su objetivo principal —leer un archivo de texto y mostrar sus líneas en orden inverso—, existen varias mejoras y extensiones que podrían implementarse en futuras versiones del proyecto. Algunas de ellas son:

1. Inversión también a nivel de palabras

Una mejora interesante sería permitir que el programa no solo invierta las líneas del archivo, sino que también invierta las palabras dentro de cada línea, siempre que contenga más de una palabra. Por ejemplo:

Entrada:

```
Hola mundo cruel  
Esta es una prueba
```

Salida:

```
prueba una es Esta  
cruel mundo Hola
```

Esto implicaría implementar funciones adicionales para dividir cada línea en palabras (strtok, arrays dinámicos), invertirlas, y luego reconstruir la línea antes de imprimirla.

2. Modo interactivo o por menú

Actualmente el programa requiere ejecutar desde la terminal o GUI. Podría implementarse un modo interactivo donde el usuario elija en tiempo real:

- Qué archivo desea procesar
- Si desea invertir líneas, palabras o ambas
- Dónde guardar el resultado (nuevo archivo o pantalla)

Esto haría el uso más flexible para usuarios no técnicos.

3. Soporte multilingüe

Agregar soporte para archivos con codificación UTF-8 ampliaría el alcance del programa, permitiendo procesar textos en otros idiomas (español, francés, árabe, etc.), incluyendo caracteres especiales y acentos.

4. Exportación en otros formatos

Una futura versión podría permitir exportar el resultado a otros formatos, como:

- `.csv` para usar en hojas de cálculo
- `.json` para consumo por aplicaciones web
- `.html` para visualización en navegadores

5. Procesamiento de múltiples archivos

Otra posible extensión es que el programa acepte **procesar varios archivos a la vez**, invirtiendo cada uno y guardando automáticamente sus resultados con nombres derivados (ej: `input1.txt` → `output1.txt`).

8. Referencias

Enumere todas las fuentes que consultaste (libros, artículos, sitios web, documentación de herramientas, etc.) utilizando un formato de citación consistente.

GTK. (n.d.). *GTK 3 Reference Manual*. <https://developer.gnome.org/gtk3/stable/>

Valgrind. (n.d.). *Valgrind Documentation*.

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall.

GeeksforGeeks. (n.d.). *Dynamic memory allocation in C using malloc(), calloc(), free() and realloc()*. <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/>

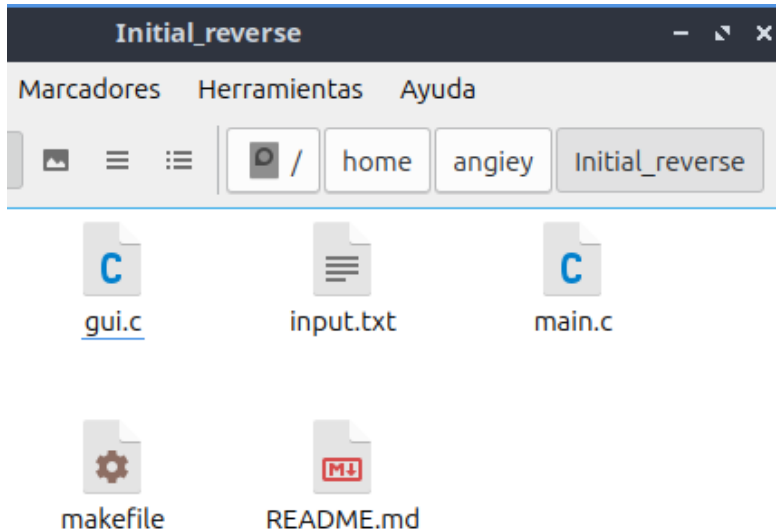
Tutorialspoint. (n.d.). C - File I/O.

https://www.tutorialspoint.com/cprogramming/c_file_io.htm

Excalidraw. (n.d.). Virtual whiteboard for sketching diagrams. <https://excalidraw.com>

9. Anexos (Opcional)

Anexo 1. Código fuente completo del proyecto



Anexo 2. Archivo [README.md](#)

```
*README.md ×
# Initial reverse

**Initial Reverse** es una aplicación escrita en lenguaje **C** que invierte el orden de las líneas de un archivo de texto.
El resultado se puede mostrar en consola o guardar en un archivo.
Además, el proyecto incluye una **interfaz gráfica (GUI) con GTK** para facilitar su uso sin necesidad de terminal.

---

## Funcionalidad principal

El programa realiza las siguientes operaciones:

✓ Lee un archivo de texto línea por línea
✓ Usa memoria dinámica con `malloc`, `getline` y `strdup`
✓ Invierte el orden de las líneas
✓ Muestra el resultado en consola o lo guarda en otro archivo
✓ Informa si el archivo está vacío o no existe
✓ Tiene una interfaz gráfica amigable con GTK

---

## 📁 Estructura del proyecto

initial_reverse/
├── main.c // Lógica del programa en consola
├── gui.c // Interfaz gráfica con GTK
├── Makefile // Automatiza compilación y ejecución
├── input.txt // Archivo de prueba
└── README.md // Este archivo
```

Requisitos

- Sistema operativo: Linux (o WSL en Windows)
- Compilador: `gcc`
- GTK 3 para interfaz gráfica:

```
```bash
sudo apt update
sudo apt install build-essential libgtk-3-dev
```
```

📦 Compilación

Compilar versión de consola: `make`

Compilar interfaz gráfica (GTK): `make gui`

▶ Ejecución

Consola: `./initial_reverse input.txt`

Guardar archivo de salida: `./initial_reverse input.txt -o salida.txt`

🖥 Interfaz Gráfica (GTK)

Ejecutar: `./gui_reverse`

Funciones de C utilizadas

`fopen()`, `fclose()` → manejo de archivos

`getline()` → lectura dinámica de líneas

`malloc()`, `realloc()`, `free()` → gestión de memoria

`strdup()` → duplicación segura de cadenas

`fprintf()` → impresión a consola o archivo

`strcmp()`, `strcat()`, `strlen()` → manejo de strings

✔ Validaciones incluidas

✕ Archivo de entrada inexistente → mensaje de error

⚠ Archivo vacío → advertencia

✓ Guardado exitoso → mensaje de confirmación

GUI con mensajes gráficos y flujo controlado

Anexo 3. Archivo main.c

```

*main.c ×
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define RED      "\033[1;31m"
#define GREEN    "\033[1;32m"
#define YELLOW   "\033[1;33m"
#define RESET    "\033[0m"

void free_lines(char **lines, int count) {
    for (int i = 0; i < count; ++i)
        free(lines[i]); // liberar cada línea
    free(lines); // liberar arreglo de punteros
}

int is_file_empty(FILE *file) {
    fseek(file, 0, SEEK_END); // Mueve el puntero al final del archivo
    long size = ftell(file);  // Obtiene la posición (tamaño en bytes)
    rewind(file);             // Vuelve al inicio del archivo
    return size == 0;
}

int main(int argc, char *argv[]) {
    // Verifica cantidad mínima de argumentos
    if (argc < 2) {
        fprintf(stderr, RED "Uso: %s <archivo_entrada> [-o archivo_salida]\n" RESET, argv[0]);
        return 1;
    }

    char *input_path = argv[1];
    char *output_path = NULL;

    // Verifica si se especificó un archivo de salida
    if (argc == 4 && strcmp(argv[2], "-o") == 0)
        output_path = argv[3];

    // Abre el archivo de entrada en modo lectura
    FILE *input_file = fopen(input_path, "r");
    if (!input_file) {
        fprintf(stderr, RED "✘ Error: No se pudo abrir el archivo de entrada: %s\n" RESET, input_path);
        return 1;
    }

    // Verifica si el archivo está vacío
    if (is_file_empty(input_file)) {
        fprintf(stderr, YELLOW "⚠ Advertencia: El archivo '%s' está vacío.\n" RESET, input_path);
        fclose(input_file);
        return 1;
    }

    // Preparación para leer líneas
    char *line = NULL;
    size_t len = 0;
    ssize_t read;

    // Arreglo dinámico de líneas con capacidad inicial de 10
    int count = 0, capacity = 10;
    char **lines = malloc(sizeof(char*) * capacity);

    if (!lines) {
        fprintf(stderr, RED "✘ Error: No se pudo asignar memoria.\n" RESET);
        fclose(input_file);
        return 1;
    }
}

```

```

// Lee línea por línea con getline y almacena en el arreglo
while ((read = getline(&line, &len, input_file)) != -1) {
    // Aumenta capacidad del arreglo si se llena
    if (count >= capacity) {
        capacity *= 2;
        char **temp = realloc(lines, sizeof(char*) * capacity);
        if (!temp) {
            fprintf(stderr, RED "✗ Error: Fallo al redimensionar memoria.\n" RESET);
            free(line);
            free_lines(lines, count);
            fclose(input_file);
            return 1;
        }
        lines = temp;
    }
    // Guarda una copia de la línea leída
    lines[count++] = strdup(line);
}

free(line); // libera el buffer de getline
fclose(input_file);

// Determina si se imprime en pantalla o se guarda en archivo
FILE *output = output_path ? fopen(output_path, "w") : stdout;
if (!output) {
    fprintf(stderr, RED "✗ Error: No se pudo abrir el archivo de salida: %s\n" RESET, output_path);
    free_lines(lines, count);
    return 1;
}

// Imprime las líneas en orden inverso
for (int i = count - 1; i >= 0; --i)
    fprintf(output, "%s", lines[i]);

// Mensaje si se guardó el resultado en archivo
if (output_path) {
    fclose(output);
    printf(GREEN "✓ Líneas invertidas guardadas en: %s\n" RESET, output_path);
}

free_lines(lines, count);
return 0;
}

```

Anexo 4. Archivo gui.c

gui.c x

```
#include <gtk/gtk.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Función para mostrar un diálogo emergente
void show_message(GtkWindow *parent, const char *message, GtkMessageType type) {
    GtkWidget *dialog = gtk_message_dialog_new(parent,
        GTK_DIALOG_MODAL,
        type,
        GTK_BUTTONS_OK,
        "%s", message);
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

// Verifica si el archivo está vacío
int is_file_empty(FILE *file) {
    fseek(file, 0, SEEK_END);
    long size = ftell(file);
    rewind(file);
    return size == 0;
}

// Invierte las líneas del archivo y las retorna como un string único
char *invert_file(const char *filename, GtkWindow *parent) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        show_message(parent, "x Error: No se pudo abrir el archivo.", GTK_MESSAGE_ERROR);
        return NULL;
    }

    if (is_file_empty(file)) {
        fclose(file);
        show_message(parent, "⚠ Advertencia: El archivo está vacío.", GTK_MESSAGE_WARNING);
        return NULL;
    }

    char **lines = NULL;
    int capacity = 10, count = 0;
    lines = malloc(capacity * sizeof(char *));
    char *line = NULL;
    size_t len = 0;

    while (getline(&line, &len, file) != -1) {
        if (count >= capacity) {
            capacity *= 2;
            lines = realloc(lines, capacity * sizeof(char *));
        }
        lines[count++] = strdup(line);
    }

    fclose(file);
    free(line);
}
```



```

size_t total_size = 0;
for (int i = 0; i < count; i++)
    total_size += strlen(lines[i]);

char *result = malloc(total_size + 1);
result[0] = '\0';

for (int i = count - 1; i >= 0; i--) {
    strcat(result, lines[i]);
    free(lines[i]);
}

free(lines);
return result;
}

// Callback del botón
void on_open_clicked(GtkButton *button, gpointer user_data) {
    GtkWidget *text_view = GTK_WIDGET(user_data);
    GtkWidget *window = gtk_widget_get_toplevel(GTK_WIDGET(button));

    GtkWidget *dialog = gtk_file_chooser_dialog_new("Seleccionar archivo",
        GTK_WINDOW(window),
        GTK_FILE_CHOOSER_ACTION_OPEN,
        "Cancelar", GTK_RESPONSE_CANCEL,
        "_Abrir", GTK_RESPONSE_ACCEPT,
        NULL);

    if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_ACCEPT) {
        char *filename = gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(dialog));

        char *content = invert_file(filename, GTK_WINDOW(window));
    }
}

```

```

        if (content) {
            GtkTextBuffer *buffer = gtk_text_view_get_buffer(GTK_TEXT_VIEW(text_view));
            gtk_text_buffer_set_text(buffer, content, -1);
            free(content);
        }

        g_free(filename);
    }

    gtk_widget_destroy(dialog);
}

int main(int argc, char *argv[]) {
    gtk_init(&argc, &argv);

    GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window), "Initial Reverse");
    gtk_window_set_default_size(GTK_WINDOW(window), 700, 500);
    g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);

    GtkWidget *vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
    GtkWidget *button = gtk_button_new_with_label("Abrir archivo");
    GtkWidget *text_view = gtk_text_view_new();
    gtk_text_view_set_editable(GTK_TEXT_VIEW(text_view), FALSE);
    gtk_text_view_set_wrap_mode(GTK_TEXT_VIEW(text_view), GTK_WRAP_WORD);

    gtk_box_pack_start(GTK_BOX(vbox), button, FALSE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(vbox), text_view, TRUE, TRUE, 0);
    gtk_container_add(GTK_CONTAINER(window), vbox);

    g_signal_connect(button, "clicked", G_CALLBACK(on_open_clicked), text_view);

    gtk_widget_show_all(window);
    gtk_main();

    return 0;
}

```

Anexo 5. Archivo makefile

makefile ×

Makefile

CC = gcc

CFLAGS = -Wall -Wextra -g

all: initial_reverse

initial_reverse: main.c

\$(CC) \$(CFLAGS) -o initial_reverse main.c

gui: gui.c

\$(CC) gui.c -o gui_reverse `pkg-config --cflags --libs gtk+-3.0`

run: all

./initial_reverse input.txt

02:

\$(CC) -O2 main.c -o initial_reverse_02

03:

\$(CC) -O3 main.c -o initial_reverse_03

test:

\$(CC) -O0 main.c -o initial_reverse_00

\$(CC) -O2 main.c -o initial_reverse_02

\$(CC) -O3 main.c -o initial_reverse_03

@echo "\n== Tiempo de ejecución (00) =="

/usr/bin/time -v ./initial_reverse_00 input.txt > /dev/null

@echo "\n== Tiempo de ejecución (02) =="

/usr/bin/time -v ./initial_reverse_02 input.txt > /dev/null

@echo "\n== Tiempo de ejecución (03) =="

/usr/bin/time -v ./initial_reverse_03 input.txt > /dev/null

valgrind:

valgrind ./initial_reverse input.txt -o salida.txt

valgrind-02:

valgrind ./initial_reverse_02 input.txt -o salida.txt

valgrind-03:

valgrind ./initial_reverse_03 input.txt -o salida.txt

show-output:

@echo "\n== Contenido de salida.txt =="

cat salida.txt

open-output:

xdg-open salida.txt

size:

@echo "\n== Tamaños de ejecutables =="

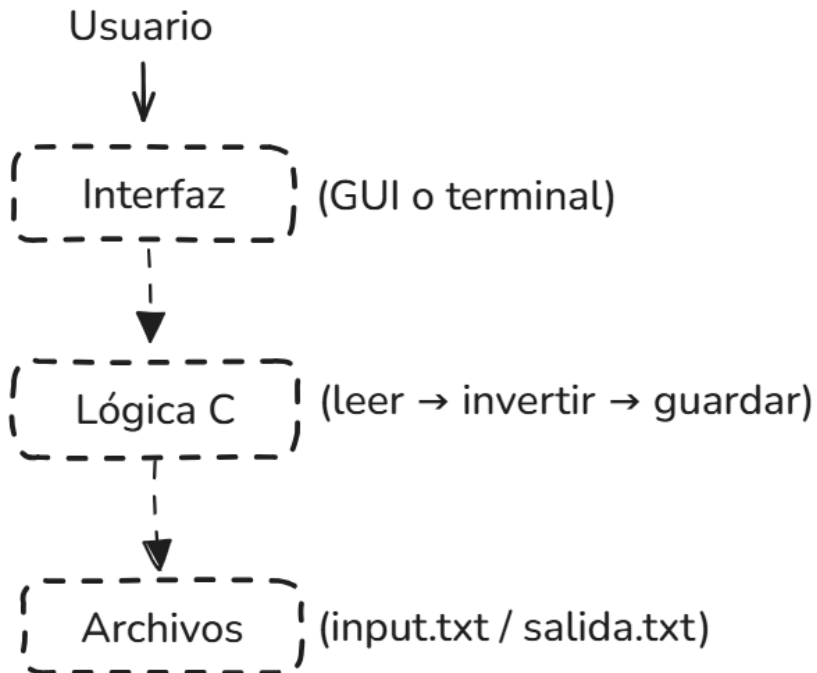
ls -lh initial_reverse_0*

clean:

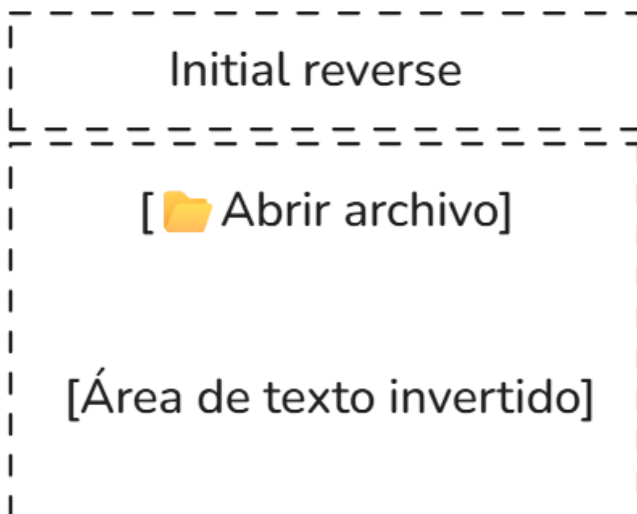
rm -f initial_reverse initial_reverse_0* gui_reverse salida.txt

Anexo 6. Diagramas detallados

Anexo 7. Diagrama de arquitectura:



Anexo 8. Diagrama de GUI (mockup)



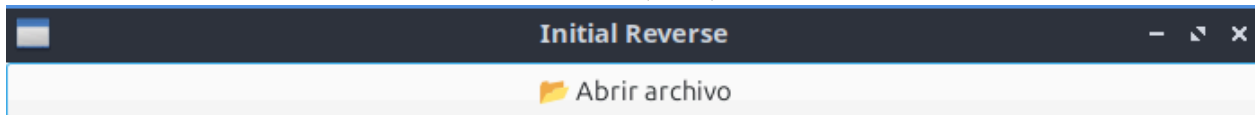
Anexo 9. Compilaciones y resultado mostrado en pantalla:

```
angiey@angie-virtualbox:~/Initial_reverse$ make
gcc -Wall -Wextra -g -o initial_reverse main.c
angiey@angie-virtualbox:~/Initial_reverse$ make gui
gcc gui.c -o gui_reverse `pkg-config --cflags --libs gtk+-3.0`
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
Esta
es
una
prueba
para
el
proyecto
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
proyecto
el
para
prueba
una
es
Esta
```

Guardamos en otro archivo (en este caso lo hemos llamado salida.txt). También, ejecutamos la interfaz gráfica (GTK)

```
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt -o salida.txt
✓ Líneas invertidas guardadas en: salida.txt
angiey@angie-virtualbox:~/Initial_reverse$ ./gui_reverse
```

Resultado de ejecución de la interfaz gráfica (GTK):



Seleccionar archivo

Recientes

Carpeta personal

Escritorio

Descargas

Documentos

Imágenes

Música

Videos

sf_CompartidaVM

Initial_reverse

VBox_GAs_7.1.6

Otras ubicaciones

| Nombre | Lugar | Tamaño | Tipo | Accedido |
|-----------|-----------------|----------|-------|----------|
| input.txt | Initial_reverse | 36 bytes | Texto | 14:30 |

Initial Reverse

Abrir archivo

proyecto
el
para
prueba
una
es
Esta

Anexo 10. Archivos de configuración.

```
makefile ×
# Makefile

CC = gcc
CFLAGS = -Wall -Wextra -g

all: initial_reverse

initial_reverse: main.c
    $(CC) $(CFLAGS) -o initial_reverse main.c

gui: gui.c
    $(CC) gui.c -o gui_reverse `pkg-config --cflags --libs gtk+-3.0`

run: all
    ./initial_reverse input.txt

02:
    $(CC) -O2 main.c -o initial_reverse_02

03:
    $(CC) -O3 main.c -o initial_reverse_03

test:
    $(CC) -O0 main.c -o initial_reverse_00
    $(CC) -O2 main.c -o initial_reverse_02
    $(CC) -O3 main.c -o initial_reverse_03
    @echo "\n== Tiempo de ejecución (00) =="
    /usr/bin/time -v ./initial_reverse_00 input.txt > /dev/null
    @echo "\n== Tiempo de ejecución (02) =="
    /usr/bin/time -v ./initial_reverse_02 input.txt > /dev/null
    @echo "\n== Tiempo de ejecución (03) =="
    /usr/bin/time -v ./initial_reverse_03 input.txt > /dev/null

valgrind:
    valgrind ./initial_reverse input.txt -o salida.txt

valgrind-02:
    valgrind ./initial_reverse_02 input.txt -o salida.txt

valgrind-03:
    valgrind ./initial_reverse_03 input.txt -o salida.txt

show-output:
    @echo "\n== Contenido de salida.txt =="
    cat salida.txt

open-output:
    xdg-open salida.txt

size:
    @echo "\n== Tamaños de ejecutables =="
    ls -lh initial_reverse_0*

clean:
    rm -f initial_reverse initial_reverse_0* gui_reverse salida.txt
```

Anexo 11. Archivo gui.c

```
gui.c x
#include <gtk/gtk.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Función para mostrar un diálogo emergente
void show_message(GtkWindow *parent, const char *message, GtkMessageType type) {
    GtkWidget *dialog = gtk_message_dialog_new(parent,
        GTK_DIALOG_MODAL,
        type,
        GTK_BUTTONS_OK,
        "%s", message);
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

// Verifica si el archivo está vacío
int is_file_empty(FILE *file) {
    fseek(file, 0, SEEK_END);
    long size = ftell(file);
    rewind(file);
    return size == 0;
}

// Invierte las líneas del archivo y las retorna como un string único
char *invert_file(const char *filename, GtkWindow *parent) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        show_message(parent, "✖ Error: No se pudo abrir el archivo.", GTK_MESSAGE_ERROR);
        return NULL;
    }

    if (is_file_empty(file)) {
        fclose(file);
        show_message(parent, "⚠ Advertencia: El archivo está vacío.", GTK_MESSAGE_WARNING);
        return NULL;
    }

    char **lines = NULL;
    int capacity = 10, count = 0;
    lines = malloc(capacity * sizeof(char *));
    char *line = NULL;
    size_t len = 0;

    while (getline(&line, &len, file) != -1) {
        if (count >= capacity) {
            capacity *= 2;
            lines = realloc(lines, capacity * sizeof(char *));
        }
        lines[count++] = strdup(line);
    }

    fclose(file);
    free(line);
}
```



```

size_t total_size = 0;
for (int i = 0; i < count; i++)
    total_size += strlen(lines[i]);

char *result = malloc(total_size + 1);
result[0] = '\0';

for (int i = count - 1; i >= 0; i--) {
    strcat(result, lines[i]);
    free(lines[i]);
}

free(lines);
return result;
}

// Callback del botón
void on_open_clicked(GtkButton *button, gpointer user_data) {
    GtkWidget *text_view = GTK_WIDGET(user_data);
    GtkWidget *window = gtk_widget_get_toplevel(GTK_WIDGET(button));

    GtkWidget *dialog = gtk_file_chooser_dialog_new("Seleccionar archivo",
        GTK_WINDOW(window),
        GTK_FILE_CHOOSER_ACTION_OPEN,
        "Cancelar", GTK_RESPONSE_CANCEL,
        "Abrir", GTK_RESPONSE_ACCEPT,
        NULL);

    if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_ACCEPT) {
        char *filename = gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(dialog));

        char *content = invert_file(filename, GTK_WINDOW(window));
    }
}

```

```

        if (content) {
            GtkTextBuffer *buffer = gtk_text_view_get_buffer(GTK_TEXT_VIEW(text_view));
            gtk_text_buffer_set_text(buffer, content, -1);
            free(content);
        }

        g_free(filename);
    }

    gtk_widget_destroy(dialog);
}

int main(int argc, char *argv[]) {
    gtk_init(&argc, &argv);

    GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window), "Initial Reverse");
    gtk_window_set_default_size(GTK_WINDOW(window), 700, 500);
    g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);

    GtkWidget *vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
    GtkWidget *button = gtk_button_new_with_label("Abrir archivo");
    GtkWidget *text_view = gtk_text_view_new();
    gtk_text_view_set_editable(GTK_TEXT_VIEW(text_view), FALSE);
    gtk_text_view_set_wrap_mode(GTK_TEXT_VIEW(text_view), GTK_WRAP_WORD);

    gtk_box_pack_start(GTK_BOX(vbox), button, FALSE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(vbox), text_view, TRUE, TRUE, 0);
    gtk_container_add(GTK_CONTAINER(window), vbox);

    g_signal_connect(button, "clicked", G_CALLBACK(on_open_clicked), text_view);

    gtk_widget_show_all(window);
    gtk_main();

    return 0;
}

```

Anexo 12. Resultados extensos de pruebas.

Tabla 1. Casos de prueba manuales

| # Prueba | Archivo de entrada | Contenido | Resultado esperado | Estado |
|----------|---------------------------|---------------------------------|--------------------------------|--------|
| 1 | input.txt | línea 1\nlínea 2 | línea 2\nlínea 1 | ☑ |
| 2 | vacío.txt | (vacío) | Advertencia
"archivo vacío" | ☑ |
| 3 | noexiste.txt | (no existe) | Error al abrir
archivo | ☑ |
| 4 | una_linea.txt | solo una línea | igual | ☑ |
| 5 | linea_varias_palabras.txt | Varias palabras
en una línea | línea 2/línea1 | ☑ |

Ejecución con 2 líneas de texto (cada 1 con una palabra):

```
angiey@angie-virtualbox: ~/Initial_reverse x
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
Sistemas
operativos
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
operativos
Sistemas
```

Ejecución con archivo de entrada vacío:

```
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
⚠ Advertencia: El archivo 'input.txt' está vacío.
angiey@angie-virtualbox:~/Initial_reverse$
```

Ejecución cuando el archivo input.txt no existe:

```
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
cat: input.txt: No existe el archivo o el directorio
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
✗ Error: No se pudo abrir el archivo de entrada: input.txt
angiey@angie-virtualbox:~/Initial_reverse$
```

Ejecución cuando el archivo input.txt tiene 1 sola línea (queda igual):

```
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
Hola
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
Hola
```

Ejecución cuando el archivo input.txt tiene más de una palabra en la línea:

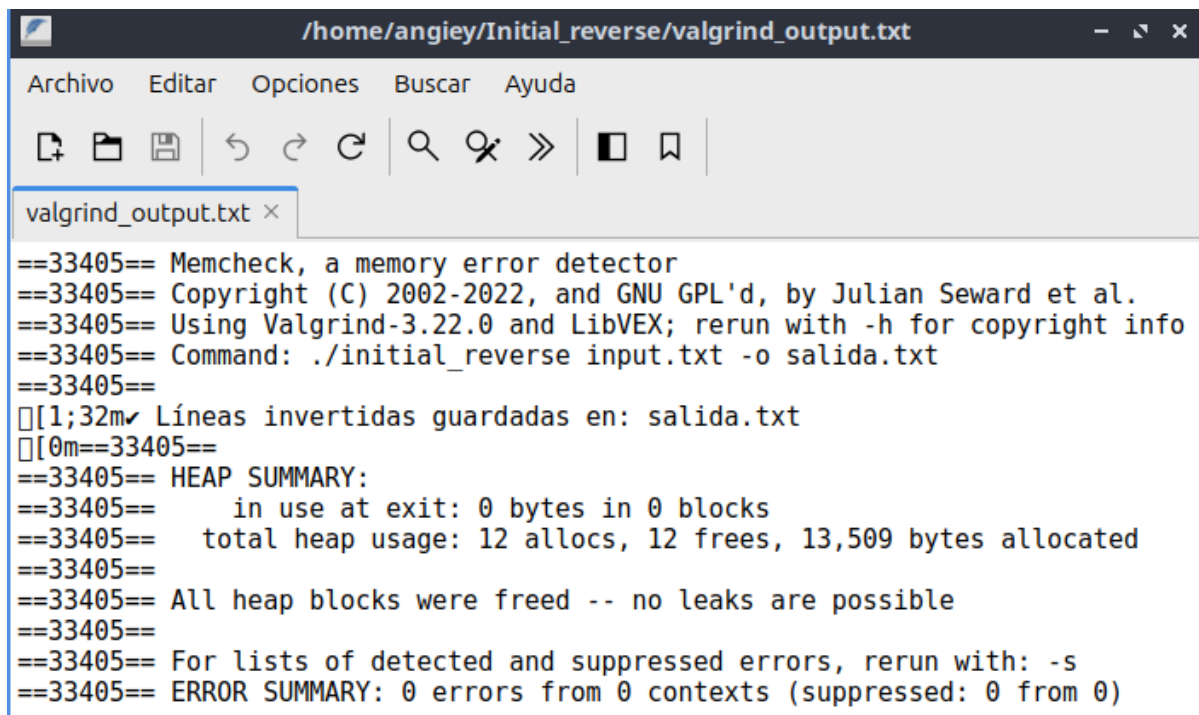
```
angiey@angie-virtualbox:~/Initial_reverse$ cat input.txt
Esta es una
prueba experimental
para la
materia de
sistemas operativos
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt
sistemas operativos
materia de
para la
prueba experimental
Esta es una
angiey@angie-virtualbox:~/Initial_reverse$ ./initial_reverse input.txt -o salida.txt
✓ Líneas invertidas guardadas en: salida.txt
```

Validación con **valgrind** (para memoria dinámica)

```
angiey@angie-virtualbox:~/Initial_reverse$ valgrind ./initial_reverse input.txt -o salida.txt
==33396== Memcheck, a memory error detector
==33396== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al
.
==33396== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==33396== Command: ./initial_reverse input.txt -o salida.txt
==33396==
✓ Líneas invertidas guardadas en: salida.txt
==33396==
==33396== HEAP SUMMARY:
==33396==    in use at exit: 0 bytes in 0 blocks
==33396==   total heap usage: 12 allocs, 12 frees, 10,437 bytes allocated
==33396==
==33396== All heap blocks were freed -- no leaks are possible
==33396==
==33396== For lists of detected and suppressed errors, rerun with: -s
==33396== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Guardamos el resultado en un archivo txt:

```
angiey@angie-virtualbox:~/Initial_reverse$ valgrind ./initial_reverse input.txt -o salida.txt &> valgrind_output.txt
```

A screenshot of a text editor window titled "/home/angiey/Initial_reverse/valgrind_output.txt". The window has a menu bar with "Archivo", "Editar", "Opciones", "Buscar", and "Ayuda". Below the menu is a toolbar with icons for file operations and editing. The main text area shows the output of a Valgrind run. The output includes version information, copyright details, the command used, a confirmation of saved lines, a heap summary, and error summary, all indicating a successful run with no leaks or errors.

```
==33405== Memcheck, a memory error detector
==33405== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==33405== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==33405== Command: ./initial_reverse input.txt -o salida.txt
==33405==
[1;32m✓ Líneas invertidas guardadas en: salida.txt
[0m==33405==
==33405== HEAP SUMMARY:
==33405==      in use at exit: 0 bytes in 0 blocks
==33405==    total heap usage: 12 allocs, 12 frees, 13,509 bytes allocated
==33405==
==33405== All heap blocks were freed -- no leaks are possible
==33405==
==33405== For lists of detected and suppressed errors, rerun with: -s
==33405== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Anexo 13. Enlace al video expositivo

<https://youtu.be/MitDBOpUteA>