Tecnológico de Costa Rica

Curso:

Arquitectura de Computadoras I

Profesor

Ronny Giovanni Garcia Ramirez

Proyecto 1:

Visualizador de Datos con Ordenamiento para Linux (Individual)

Alumna:

Kendy Raquel Arias Ortiz

Fecha:

22/09/2025

Ficha Técnica del Proyecto

Tecnológico de Costa Rica
Curso: Arquitectura de Computadoras I
Profesor: Ronny Giovanni García Ramírez
Proyecto 1: Visualizador de Datos con Ordenamiento para Linux (Individual)
Alumna: Kendy Raquel Arias Ortiz
Fecha: 22/09/2025

1. Descripción del Problema

El proyecto consiste en el desarrollo de un programa en ensamblador x86 (NASM) que lea, procese y visualice datos de inventario en un entorno Linux. El sistema debe:

- Leer datos de productos desde un archivo de texto (inventario.txt), que contiene nombres y cantidades.
- Ordenar los productos alfabéticamente en memoria.
- Mostrar un gráfico de barras en la terminal, donde cada barra representa la cantidad de un producto.
- Aplicar configuraciones externas (color de barra, color de fondo y carácter utilizado) definidas en un segundo archivo (config.ini).

El desafío principal es implementar todo el flujo en lenguaje ensamblador, manipulando directamente las llamadas al sistema de Linux, sin bibliotecas de alto nivel. Este enfoque permite un control total de la memoria, la entrada/salida y la lógica del programa, a la vez que demuestra el potencial del ensamblador para aplicaciones prácticas.

2. Objetivo General

Crear un programa modular y flexible que:

- Procesar archivos de texto en tiempo de ejecución.
- Ordene información alfabéticamente.
- Genere una representación visual personalizable dentro de la terminal Linux.

El uso de Git y GitHub para el control de versiones asegura un desarrollo organizado y documentado.

3. Archivos de Entrada

El programa trabaja con dos archivos principales, que pueden modificarse para generar salidas diferentes:

- **inventario.txt**: Contiene los productos y sus cantidades, por ejemplo:
  manzanas:12
  peras:8
  naranjas:25
  kiwis:5

- **config.ini**: Define el carácter de las barras y los códigos ANSI de color. Ejemplo:
  caracter_barra:█
  color_barra:92
  color_fondo:40

  Donde 92 representa el color verde brillante y 40 el fondo negro. Cambiando estos valores se pueden obtener gráficos con diferentes estilos y colores sin modificar el código.

4. Flujo del Programa

El programa se desarrolla en cuatro etapas principales:

4.1 Lectura de configuración (config.ini)

Se abre el archivo de configuración usando syscalls de Linux (open, read, close). Los parámetros leídos —caracter_barra, color_barra y color_fondo— se almacenan en memoria y controlan la apariencia del gráfico final. Gracias a este diseño, la personalización no requiere recompilar el programa.

4.2 Lectura de inventario (inventario.txt)

Se abre el archivo de inventario, leyendo línea por línea. Cada línea se analiza para separar el nombre del producto de la cantidad numérica, que se guarda en una estructura de memoria. La separación se realiza detectando el carácter delimitador “:”.

4.3 Ordenamiento de datos

Los productos almacenados se ordenan alfabéticamente mediante **Bubble Sort**, elegido por su simplicidad en entornos de bajo nivel. El algoritmo recorre la lista, comparando nombres carácter por carácter, e intercambia posiciones cuando detecta desorden. Aunque tiene complejidad O(n2), es ideal para el tamaño reducido de los datos.

4.4 Visualización en la terminal

Por cada producto ordenado, el programa imprime: el nombre del producto, una barra de color, formada repitiendo el caracter_barra tantas veces como la cantidad indicada. Al final de la línea se imprime la cantidad numérica.
Los colores y el estilo se aplican utilizando códigos de escape ANSI, que permiten controlar tanto el color del texto como el del fondo. El resultado es un gráfico de barras dinámico dentro de la terminal, que facilita la interpretación visual de los datos.

5. Diagrama de Flujo del Programa

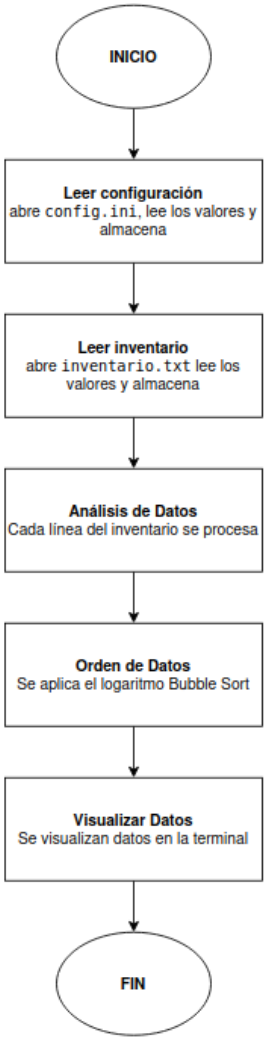

Diagrama de Flujo del Programa

6. Algoritmos y Técnicas Utilizadas

6.1 Lectura de archivos

- Syscalls (open, read, close) para manipular los archivos de texto.
- Procesamiento de cadenas en memoria para separar claves y valores.
- Uso de buffers para almacenar temporalmente los datos leídos.

6.2 Ordenamiento (Bubble Sort)

- Comparación carácter por carácter de los nombres de producto.
- Intercambio de posiciones en memoria cuando el orden alfabético no se cumple.
- **Ventaja:** implementación simple y comprensible en ensamblador.

6.3 Visualización con códigos ANSI

- Impresión repetitiva del carácter de barra mediante bucles.
- Aplicación de secuencias ANSI para modificar el color del texto y el fondo.
- **Resultado final:** un gráfico personalizable sin dependencias externas.

7. Pruebas y Ejecución

Se realizaron varias pruebas cambiando los valores en config.ini. Por ejemplo:

- **Configuración verde:** caracter_barra:█, color_barra:92, color_fondo:40.
- **Configuración azul:** caracter_barra:*, color_barra:94, color_fondo:40.

En cada caso, el programa generó correctamente un gráfico ordenado, reflejando las nuevas configuraciones sin necesidad de recompilación.
El proceso necesario para verificar el correcto funcionamiento del programa y reproducir los resultados es el siguiente:

7.1 Preparación del entorno

- **Sistema Operativo:** Linux (probado en Ubuntu 22.04).
- **Dependencias:** Ensamblador NASM y enlazador ld.
- **Archivos requeridos:**
    - inventario_final.asm (programa principal).
    - config.ini (parámetros de configuración).
    - inventario.txt (datos de inventario).

7.2 Compilación

Abrir una terminal en la carpeta src y ejecutar:

nasm -f elf64 inventario_final.asm -o inventario_final.o
ld inventario_final.o -o inventario_final

Esto genera el ejecutable inventario_final.

7.3 Ejecución

En la misma carpeta ejecutar:
./inventario_final

El programa leerá los archivos config.ini e inventario.txt, ordenará los datos alfabéticamente y mostrará el gráfico en la terminal.

7.3 Ejecución

A continuación se muestran dos ejemplos de pruebas diferentes para el inventario y dos para el config:

7.3.1 Prueba 1

Con una entrada de inventario.txt de:
manzanas:30
peras:8
naranjas:25
kiwis:5
albaricoque:15

Y de config.ini:
caracter_barra:
color_barra:92
color_fondo:40

Se da una respuesta en la terminal de:

```
clear@clear-System-Product-Name:~/Documents/Arquitectura-de-Computadoras-VDOL-Ar
ias-Ortiz/src$ ./inventario_final
albaricoque:  15
kiwis:  5
manzanas:  30
naranjas:  25
peras:  8
```

Cumpliendo con lo estipulado. Se respetan tanto los colores como el símbolo elegido.

7.3.1 Prueba 2

Con una entrada de inventario.txt de:
bananos:6
sandias:3
manzanas:1
mamones:3
piñas:1

Y de config.ini:
caracter_barra:*
color_barra:94
color_fondo:40

Se da una respuesta en la terminal de:

```
clear@clear-System-Product-Name:~/Documents/Arquitectura-de-Computadoras-VDOL-Ar
ias-Ortiz/src$ ./inventario_final
bananos: ******6
mamones: ***3
manzanas: *1
piñas: *1
sandias: ***3
```

Cumpliendo, nuevamente con lo estipulado. Se respetan tanto los colores como el símbolo elegido.

8. Repositorio y Control de Versiones

El proyecto fue desarrollado utilizando **Git** para el control de versiones y alojado en **GitHub**. El historial de commits refleja:

- Avances progresivos en el código.
- Pruebas de diferentes configuraciones.
- Limpieza de archivos intermedios y binarios para mantener un repositorio profesional.

9. Conclusiones

Este proyecto demuestra que el lenguaje ensamblador puede utilizarse para tareas de procesamiento de datos y generación de gráficos en consola, combinando control de bajo nivel con flexibilidad visual.

Entre los principales aprendizajes destacan:

- Manipulación directa del sistema operativo mediante syscalls.
- Implementación de algoritmos clásicos (Bubble Sort) en un entorno de bajo nivel.
- Importancia de la modularidad para permitir configuraciones externas.
- Uso de Git y GitHub como herramientas esenciales de trabajo colaborativo y documentación del progreso.

10. Referencias

- **Algoritmos y Estructuras de Datos:**
  - **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009).** ***Introduction to Algorithms*** **(3rd ed.). The MIT Press.** (Referencia al capítulo sobre algoritmos de ordenamiento simple, como el Bubble Sort).
- **Página oficial de NASM:** Para la documentación completa del ensamblador NASM.
  - `https://www.nasm.us/docs.html`
- **Secuencias de escape ANSI:** Para una explicación detallada de los códigos de color y formato en la terminal.
  - `https://en.wikipedia.org/wiki/ANSI_escape_code`