UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

UNAH-CAMPUS COMAYAGUA

CARRERA:

INGENIERÍA EN SISTEMAS

ASIGNATURA:

IS-412
SISTEMAS OPERATIVOS I

CATEDRÁTICO:

ELMER DANUARY PADILLA SORTO

ESTUDIANTES:

CUENTA	NOMBRE
20231900189	JORGE ABRAHAM FAJARDO LÓPEZ
20231900184	RICHARD ORLANDO ANDINO VILLANUEVA

FECHA ENTREGA:

24/ABRIL/2025

1. Introducción

El presente informe documenta el desarrollo de un simulador de algoritmos de planificación de procesos para sistemas operativos, implementado en Python con la librería Tkinter para la interfaz gráfica. El proyecto fue desarrollado en Visual Studio Code como entorno de desarrollo integrado.

2. Descripción del Proyecto

El simulador permite analizar y comparar el comportamiento de diferentes algoritmos de planificación de procesos bajo diversas condiciones, proporcionando tanto visualizaciones gráficas como métricas cuantitativas de desempeño.

3. Funcionalidades Implementadas

3.1 Gestión de Procesos

- Ingreso manual: Mediante formularios en la interfaz gráfica

- Carga por archivo: Soporte para importar procesos desde archivos JSON

- Campos por proceso:

  - ID único

  - Tiempo de llegada

  - Tiempo de ráfaga (CPU)

  - Prioridad (para algoritmos que la requieren)

3.2 Algoritmos Implementados

1. FCFS (First Come First Served): Planificación por orden de llegada

2. Round Robin: Con quantum configurable por el usuario

3. SJF (Shortest Job First): Tanto en versión expropiativa como no expropiativa

4. Por Prioridades: Implementación opcional adicional

3.3 Visualización

- Diagrama de Gantt: Representación gráfica de la secuencia de ejecución

- Métricas de desempeño: Presentación tabular de resultados

- Interfaz intuitiva: Organizada en pestañas para diferentes funcionalidades

3.4 Métricas Calculadas

- Tiempo de espera promedio

- Tiempo de respuesta promedio

- Tiempo de retorno promedio

- Porcentaje de uso de CPU

4. Implementación Técnica

4.1 Estructura del Código

/proyecto/

├─── algoritmos.py # Algoritmos de planificación (FCFS, RR, SJF, etc.)

├─── main.py # Punto de entrada de la aplicación con GUI Tkinter

├─── utils.py # Funciones de utilidad para métricas, carga, etc.

└─── __pycache__/ # Archivos temporales generados por Python

4.2 Librerías Utilizadas

- Tkinter: Para la interfaz gráfica principal

- Matplotlib: Para generación del diagrama de Gantt

- JSON: Para serialización de datos

- Time: Para simulaciones en tiempo real (opcional)

5. Manual de Usuario

5.1 Requisitos del Sistema

- Python 3.8 o superior

- Librerías especificadas en requirements.txt

5.2 Instalación

```bash

pip install -r requirements.txt

```

5.3 Uso Básico

1. Ingresar procesos manualmente o cargar desde archivo

2. Seleccionar algoritmo de planificación

3. Configurar parámetros específicos (quantum para RR)

4. Ejecutar simulación

5. Analizar resultados en pestañas de Gantt y métricas

6. Resultados y Conclusiones

6.1 Hallazgos

- Cada algoritmo muestra comportamientos característicos ante diferentes cargas de trabajo

- La visualización gráfica ayuda a comprender las diferencias entre algoritmos

- Las métricas cuantitativas permiten comparación objetiva

6.2 Mejoras Futuras

- Implementar algoritmos adicionales (SRT, Multilevel Queue)

- Permitir animación del diagrama de Gantt

- Añadir modo de comparación lado a lado

- Exportar resultados a formatos PDF/Excel

7. Conclusiones

El simulador desarrollado cumple con los objetivos planteados, proporcionando una herramienta educativa para entender el comportamiento de diferentes algoritmos de planificación. La implementación con Tkinter ofrece una interfaz accesible sin requerir dependencias externas complejas.