**UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS**

**UNAH – CURC**

**CARRERA:**

INGENIERÍA EN SISTEMAS

**TEMA:**

SIMULACIÓN DE ALGORITMOS DE PLANIFICACIÓN DE PROCESOS

**CLASE:**

IS412 – SISTEMAS OPERATIVOS I

**CATEDRÁTICO:**

ING. ELMER PADILLA

**ALUMNA:**

DINORA ROXANA PADILLA ORTIZ

**CUENTA:**

20181900411

**COMAYAGUA, HONDURAS**

**INDICE**

**Portada 1**

**Índice** 2

**Introducción** 3

**Objetivos4**

Objetivos Generales4

Objetivos Específicos4

**Tecnologías Utilizadas5**

**Estructura del Sistema 6**

**Desarrollo del Sistema 7**

**Algoritmos Implementados8**

**Cálculo de Métricas 8**

**Visualización con diagrama de Gantt 9**

**Pruebas Realizadas 10**

**Conclusiones 13**

**INTRODUCCIÓN**

En los sistemas operativos, la planificación de procesos es una tarea fundamental que determina el orden en que los procesos serán ejecutados por la CPU. Esta decisión tiene un impacto directo en el rendimiento del sistema, la eficiencia de los recursos y la experiencia del usuario.

Este proyecto consiste en el desarrollo de un simulador gráfico de planificación de procesos utilizando el lenguaje de programación Java y la biblioteca Swing para la interfaz gráfica. El simulador permite al usuario ingresar procesos de forma manual o cargarlos desde un archivo de texto plano. Posteriormente, puede aplicar distintos algoritmos de planificación como FCFS (First Come First Served), SJF (Shortest Job First) y Round Robin, observando el comportamiento de los procesos en una tabla de métricas y en un diagrama de Gantt visual.

El desarrollo de este simulador tiene como finalidad fortalecer el aprendizaje práctico de los conceptos teóricos de planificación, permitiendo al estudiante interactuar con los algoritmos y visualizar en tiempo real los efectos de sus decisiones en la CPU.

**OBJETIVOS**

**Objetivo general**

Desarrollar un simulador de planificación de procesos en Java que permita al usuario visualizar el comportamiento de los algoritmos FCFS, SJF y Round Robin, así como obtener métricas de rendimiento y representar la ejecución mediante un diagrama de Gantt.

**Objetivos específicos**

* Implementar una interfaz gráfica amigable para el ingreso y visualización de procesos.
* Incorporar los algoritmos de planificación FCFS, SJF y Round Robin con cálculo de métricas.
* Permitir la carga de procesos desde archivos .txt estructurados.
* Generar un diagrama de Gantt que represente visualmente la secuencia de ejecución.
* Calcular los tiempos de espera, respuesta y retorno, así como sus promedios.
* Permitir la exportación de resultados a archivos.

**TECNOLOGÍAS UTILIZADAS**

El simulador fue desarrollado utilizando las siguientes herramientas:

* Java 16.02: Lenguaje de programación orientado a objetos, ampliamente utilizado para el desarrollo de aplicaciones de escritorio.
* Apache NetBeans IDE 16: Entorno de desarrollo integrado (IDE) utilizado para escribir, compilar y probar el código Java.
* Swing: Biblioteca gráfica integrada en Java para el diseño de interfaces gráficas (GUI).
* JTable y JPanel: Componentes visuales de Swing usados para mostrar los procesos en una tabla y representar gráficamente el diagrama de Gantt.
* Archivos .txt: Formato de texto plano utilizado para cargar listas de procesos desde un archivo externo.
* Archivos .csv: Formato utilizado para exportar los resultados de las simulaciones.

**ESTRUCTURA DEL SISTEMA**

* El sistema está compuesto por los siguientes paquetes y clases principales:

***Paquete modelo***

* Proceso.java: Clase que representa a un proceso con sus atributos principales: ID, tiempo de llegada, ráfaga, prioridad y métricas calculadas (espera, respuesta, retorno).

***Paquete algoritmos***

* FCFS.java: Implementa el algoritmo First Come First Served, procesando en orden de llegada.
* SJF.java: Implementa Shortest Job First, dando prioridad a los procesos con ráfagas más cortas.
* RoundRobin.java: Implementa Round Robin, utilizando un quantum configurable y asignando turnos circulares a los procesos.

***Paquete gui***

* VentanaSimulador.java: Ventana principal del simulador. Permite el ingreso manual de procesos, la carga desde archivo, la ejecución de algoritmos, la visualización del Gantt y el guardado de resultados.
* PanelGantt.java: Componente visual personalizado que representa gráficamente la ejecución de los procesos utilizando colores únicos por proceso.

**DESARROLLO DEL SISTEMA**

***Ingreso manual y desde archivo .txt***

El simulador permite ingresar procesos de forma manual a través de campos de texto donde el usuario introduce el ID, el tiempo de llegada, la ráfaga de CPU y la prioridad. También se puede cargar un archivo .txt con una lista de procesos ya definidos.

El formato del archivo debe tener la siguiente estructura:

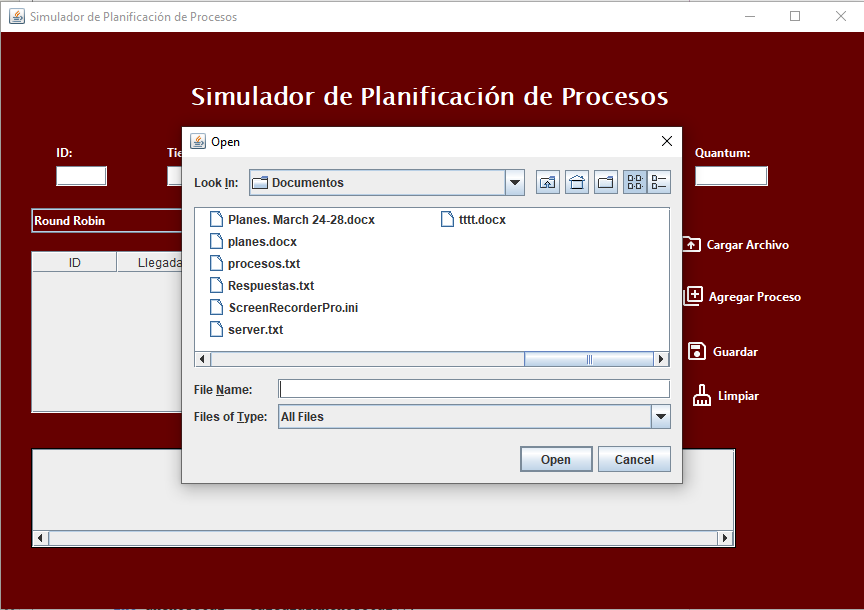
ID, Llegada, Ráfaga, Prioridad

P1,0,5,2

P2,2,3,1

P3,4,2,3

P4,6,1,1

Al presionar el botón “Cargar desde archivo”, los datos son leídos línea por línea, validados y agregados automáticamente a la tabla del simulador.

**ALGORITMOS IMPLEMENTADOS**

* **FCFS (First Come First Served)**

Este algoritmo ordena los procesos por su tiempo de llegada y los ejecuta secuencialmente, uno después de otro. Calcula los tiempos de espera, respuesta y retorno, y genera una secuencia visual que incluye tiempos de inactividad si los hay.

* **SJF (Shortest Job First)**

Selecciona el proceso con la ráfaga más corta entre los que ya han llegado. Utiliza una cola de prioridad para decidir el orden de ejecución. También considera inactividad cuando no hay procesos listos para ejecutarse.

* **Round Robin**

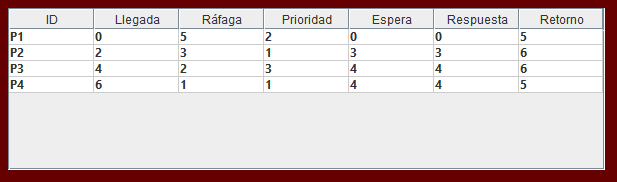
Este algoritmo reparte el tiempo de CPU entre los procesos usando un quantum definido por el usuario. Si un proceso no termina en ese tiempo, se pausa y vuelve a la cola de ejecución. Permite simular sistemas multitarea de forma justa y equitativa.

**CÁLCULO DE MÉTRICAS**

Para cada proceso, el sistema calcula automáticamente las siguientes métricas:

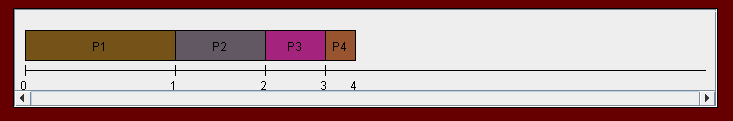
* **Tiempo de espera:** tiempo total que el proceso pasó esperando en la cola.
* **Tiempo de respuesta:** tiempo desde que llega hasta que se ejecuta por primera vez.
* **Tiempo de retorno:** tiempo total desde que llega hasta que termina.

También se calculan y muestran los promedios de estas métricas al final de cada simulación.

**VISUALIZACIÓN CON DIAGRAMA DE GANTT**

El simulador incluye un componente visual personalizado (PanelGantt) que dibuja la ejecución de los procesos en una línea de tiempo. Cada proceso tiene un color único, y los bloques se ajustan en tamaño según la duración de ejecución. Los espacios de inactividad se muestran como bloques grises ("Idle").

* Esto permite al usuario visualizar claramente:
* Qué proceso se ejecutó y en qué momento
* Cuánto tiempo duró cada ejecución
* Si hubo periodos de inactividad de la CPU

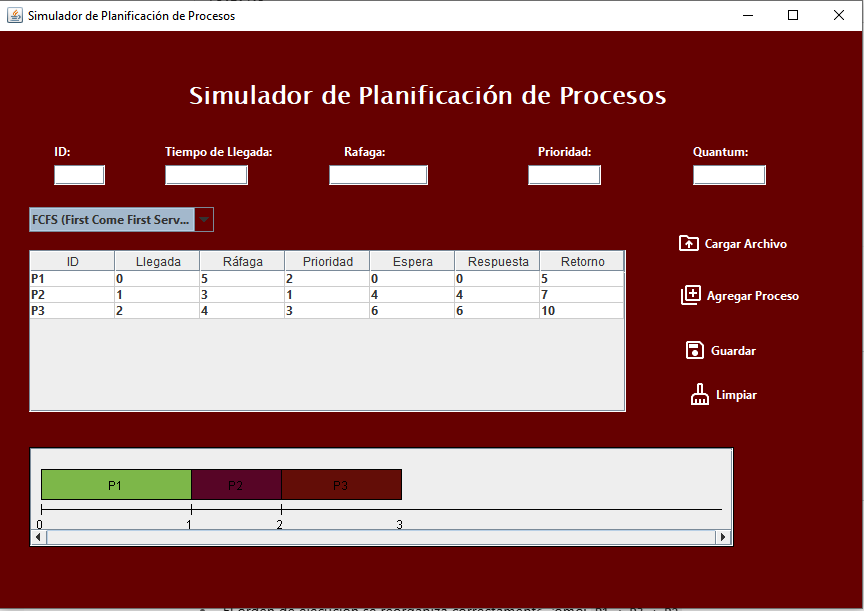


**EXPORTACIÓN DE RESULTADOS**

Una vez realizada la simulación, los resultados pueden exportarse a un archivo .csv, que incluye los datos originales del proceso y las métricas calculadas. Esto permite conservar un registro o analizar los datos en herramientas externas.

**PRUEBAS REALIZADAS**

A continuación, se describen algunas pruebas realizadas sobre el simulador de planificación, con distintos algoritmos y escenarios, para comprobar su funcionamiento, precisión en los cálculos y visualización gráfica.

**Prueba 1: FCFS con procesos consecutivos**

Algoritmo: FCFS

Observaciones:

* Los procesos se ejecutan en el orden de llegada.
* La CPU no queda inactiva.
* El diagrama de Gantt refleja correctamente la secuencia: P1 → P2 → P3
* Los tiempos de espera, respuesta y retorno se calcularon correctamente.

**Prueba 2: SJF con ráfagas variadas**

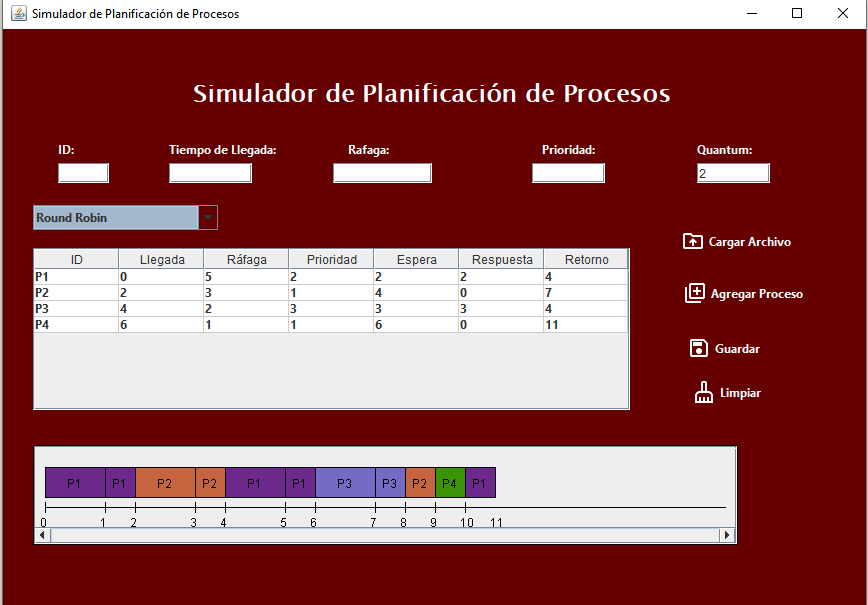


Algoritmo: SJF

Observaciones:

* El proceso P1 comienza, pero cuando P2 y P3 llegan, se prioriza el de menor ráfaga.
* El orden de ejecución se reorganiza correctamente como: P1 → P2 → P3
* Se muestra un bloque de "Idle" si ningún proceso está disponible al inicio.
* El cálculo de métricas fue correcto y el orden de ejecución fue el esperado.

**Prueba 3: Round Robin con quantum 2**

Procesos:

Algoritmo: Round Robin (Quantum = 2)

Observaciones:

* Cada proceso recibe un tiempo limitado de ejecución.
* Los procesos se intercalan en la secuencia: P1 → P2 → P1 → P3 → P2 → P4 → P1
* El Gantt refleja perfectamente el comportamiento circular.
* El simulador permitió visualizar claramente el reparto justo del CPU.

**CONCLUSIONES**

El desarrollo del simulador de planificación de procesos permitió aplicar de forma práctica los conceptos teóricos vistos en la materia de Sistemas Operativos I, específicamente los relacionados con la gestión del CPU y los algoritmos de planificación.

Mediante la implementación de algoritmos como FCFS, SJF y Round Robin, se logró comprender el comportamiento del sistema bajo distintos esquemas de asignación de tiempo, así como las diferencias entre cada uno en cuanto a eficiencia, equidad y tiempos de respuesta.

La posibilidad de ingresar procesos manualmente o desde archivo, visualizar la ejecución mediante un diagrama de Gantt, y exportar los resultados a un archivo .csv, otorgan al simulador un enfoque tanto educativo como práctico.

Además, este proyecto fortaleció habilidades de programación orientada a objetos en Java, diseño de interfaces gráficas con Swing, y manipulación de estructuras de datos como listas, colas y mapas.

El resultado es una herramienta funcional, amigable y visual, útil tanto para estudiantes como docentes que deseen explicar o analizar algoritmos de planificación de CPU.