Universidad Católica Andres Bello

Facultad de Ingeniería

Escuela de Informática

Sistemas Operativos

Prof. Gustavo Lara

Manual de Usuario: Simulador de Algoritmos de Planificación de CPU

Estudiantes:

Adrian Berbesi

Victor Chávez

Cristian De Sousa

Caracas, 12 de noviembre de 2026

Introducción

Descripción General

El Simulador de Algoritmos de Planificación de CPU es una herramienta de software desarrollada en Python, diseñada para modelar, ejecutar y analizar el comportamiento de los distintos algoritmos de gestión de procesos en un sistema operativo monoprocesador.

Este proyecto se enmarca dentro de la Unidad 4 de la asignatura Sistemas Operativos ("Administración del Procesador"), con el objetivo de proporcionar una visualización clara de cómo el kernel decide qué proceso debe ejecutarse en cada instante de tiempo. El simulador permite comparar métricas de rendimiento críticas, facilitando la comprensión de conceptos teóricos complejos mediante la práctica experimental.

Objetivos de la Herramienta

El simulador ha sido construido para cumplir con los siguientes propósitos:

- **Simulación Fidedigna:** Implementar la lógica exacta de los algoritmos FCFS, SJF (No Apropiativo), Round Robin, Prioridades (Apropiativo y No Apropiativo) y SRTF.
- **Análisis de Rendimiento:** Calcular métricas fundamentales como el Turnaround Time (Tiempo de Retorno), Waiting Time (Tiempo de Espera) y Response Time (Tiempo de Respuesta).
- **Comparativa Automatizada:** Ejecutar múltiples algoritmos sobre la misma carga de trabajo (dataset) para determinar cuál ofrece el mejor rendimiento bajo criterios específicos.
- **Visualización:** Generar diagramas de Gantt en formato textual para inspeccionar la línea de tiempo de ejecución de la CPU.

Alcance

La aplicación funciona mediante una interfaz de línea de comandos (CLI), asegurando compatibilidad con cualquier sistema operativo que soporte Python 3. El sistema permite la carga de procesos mediante archivos de configuración (JSON) predefinidos o la entrada manual de datos, ofreciendo flexibilidad tanto para pruebas rápidas como para casos de estudio complejos.

Instalación y Configuración

Requisitos del Sistema

Para ejecutar el simulador, el equipo debe cumplir con los siguientes requisitos mínimos:

- **Sistema Operativo:** Windows 10/11, macOS 10.15+, o cualquier distribución de Linux (Ubuntu, Fedora, etc.).
- **Lenguaje:** Python 3.8 o superior.
- **Dependencias:** El proyecto utiliza exclusivamente la Librería Estándar de Python y la librería Colorama (pip install colorama), esta última se debe descargar para garantizar una ejecución limpia y sin conflictos de versiones.

Estructura del Proyecto

Antes de la instalación, verifique que la carpeta del proyecto contenga la siguiente estructura de archivos, necesaria para el correcto funcionamiento de los módulos:

```
/cpu_scheduler
├── main.py              # Archivo ejecutable principal
├── core/                # Clases base (Scheduler, Timeline)
   └── algorithms/       # Lógica de los algoritmos (FCFS, SJF, RR, etc.)
├── metrics/             # Métricas
├── models/              # Modelos de datos (Process)
├── ui/                  # Interfaz de usuario y visualización
└── utils/               # Manejo de archivos y generadores
/tests
└── cases.json           # Archivo con sets de prueba
```

Pasos de Instalación

1. **Descarga del Código Fuente:** Obtenga el paquete de archivos provisto (formato .zip) y descomprímalo en una ubicación accesible de su equipo (ejemplo: Documentos/SimuladorCPU).
2. **Verificación de Python:** Abra su terminal (Símbolo del sistema en Windows o Terminal en Mac/Linux) y ejecute el siguiente comando para verificar su versión de Python:

Bash

python --version

Nota: Si el comando retorna un error, asegúrese de tener Python instalado y agregado al PATH del sistema.

Ejecución Inicial: Navegue hasta la carpeta del proyecto desde la terminal:

Bash

cd ruta/a/tu/carpeta/proyecto_so

Ejecute el programa principal para verificar la integridad de los archivos:

Bash

python main.py

Si la instalación es correcta, verá el menú de bienvenida del simulador.

Guía de Uso

Esta sección detalla el flujo de trabajo estándar para operar el simulador, desde la carga de datos hasta la interpretación de resultados.

Inicio del Simulador

Al ejecutar el comando python main.py, el sistema desplegará el encabezado institucional y verificará la existencia del archivo de casos de prueba.

Selección de Fuente de Procesos

El primer paso es definir cómo se ingresarán los procesos al sistema. El menú presentará dos opciones:

1. **Cargar desde JSON (Opción 1):**
    - Lee el archivo tests/cases.json.
    - El sistema listará los "Sets" disponibles (por ejemplo: set1, set2, set_personal).
    - El usuario debe escribir el nombre exacto del set que desea simular.
    - Recomendado para pruebas repetibles y comparativas complejas.
2. **Creación Manual (Opción 2):**
    - Permite ingresar procesos uno a uno por consola.
    - Para cada proceso, el sistema solicitará:
        - **ID:** Identificador (ej. "P1"). Dejar vacío para finalizar la carga.
        - **Tiempo de llegada:** Entero no negativo (>= 0).
        - **Tiempo de ráfaga (Burst):** Entero positivo (> 0).
        - **Prioridad:** Entero no negativo (>= 0) (Menor valor indica mayor prioridad).

Selección del Algoritmo

Una vez cargados los procesos, se mostrará una lista con sus atributos. A continuación, debe seleccionar el algoritmo de planificación:

- **1. FCFS (First Come, First Served):** Orden de llegada.
- **2. SJF (Shortest Job First):** No apropiativo, basado en la ráfaga más corta.

- **3. Round Robin:** Solicitará un valor de Quantum (tiempo máximo de ejecución por turno). El valor por defecto es 4.
- **4. Prioridades:** Solicitará si desea la versión preemptiva (apropiativa) o no preemptiva.
- **5. SRTF (Shortest Remaining Time First):** Versión apropiativa de SJF.
- **6. TODOS (Comparativa):** Ejecuta secuencialmente los 5 algoritmos anteriores sobre el mismo set de datos y genera una tabla comparativa final.

Interpretación de Resultados

Al finalizar la simulación, el sistema mostrará tres secciones de información:

A. Diagrama de Gantt (Textual)

Representación lineal del uso de la CPU.

- Formato: [ID_Proceso | inicio -> fin]
- Ejemplo: [P1 | 0 -> 5] [P2 | 5 -> 8]
- Si la CPU está inactiva, se mostrará como [None | t1 -> t2].

B. Métricas por Proceso

Una tabla detallada con los tiempos calculados para cada proceso:

- **Turnaround:** Tiempo total desde que el proceso llega hasta que termina.
- **Waiting (Espera):** Tiempo total que el proceso pasó en la cola de listos.
- **Response (Respuesta):** Tiempo desde la llegada hasta que obtuvo la CPU por primera vez.

C. Métricas del Sistema

Promedios globales útiles para evaluar la eficiencia del algoritmo:

- Promedio de Turnaround.
- Promedio de Espera.
- Promedio de Respuesta.
- Utilización de la CPU (%).

D. Comparativa y Conclusión Automática (Exclusivo Opción 6)

Si se seleccionó la opción **"6. TODOS"**, al final del reporte se mostrará un análisis agregado:

- **Tabla Comparativa:** Un cuadro resumen que pone frente a frente los promedios (Turnaround, Espera, Respuesta) y la utilización de CPU de todos los algoritmos ejecutados.
- **Conclusión Automática:** El sistema evalúa internamente los resultados y muestra un mensaje indicando cuál fue el algoritmo más eficiente para ese conjunto de datos específico (basado principalmente en minimizar el tiempo de espera promedio).

Ejemplos Prácticos

A continuación, se presentan dos casos de uso para ilustrar el funcionamiento del sistema.

Caso 1: Comparativa Automática (Uso de JSON)

Escenario: El usuario desea comparar qué algoritmo es más eficiente para el conjunto de datos set1 definido en cases.json.

1. **Ejecución:** python main.py
2. **Selección de Datos:**
    - El usuario selecciona 1 (Archivo JSON).
    - Ante la lista de sets disponibles, escribe: set1.
3. **Selección de Algoritmo:**
    - El usuario selecciona 6 (TODOS).
    - Configura Quantum = 4.
    - Configura Prioridades Preemptivo = s (Sí).

Salida del Sistema: El simulador ejecutará los 5 algoritmos uno tras otro. Al final, mostrará una tabla similar a esta:

Plaintext

Comparación entre algoritmos (promedios):

Algoritmo	Avg Turnaround	Avg Espera	Avg Respuesta	CPU Util (%)
FCFS	15.25	8.75	8.75	100.00
SJF (no apropiativo)	14.25	7.75	7.75	100.00
Round Robin	18.25	11.75	4.50	100.00
Prioridades	13.00	6.50	4.25	100.00
SRTF	13.00	6.50	4.50	100.00

Conclusión automática: mejor algoritmo para este caso es Prioridades

Análisis: El sistema identifica correctamente que Prioridades minimiza el tiempo de espera promedio para este lote específico de procesos.

Caso 2: Simulación Manual de Round Robin

Escenario: Se desea probar el comportamiento de Round Robin con un Quantum pequeño (Q=2) para dos procesos conflictivos.

1. **Datos de Entrada:**
    - **P1:** Llegada 0, Ráfaga 5.
    - **P2:** Llegada 1, Ráfaga 4.
2. **Configuración:**
    - Opción 2 (Manual).
    - Ingreso de datos P1 y P2.
    - Algoritmo 3 (Round Robin).
    - Quantum 2.

Resultado (Diagrama de Gantt):

[P1 | 0 -> 2] <- P1 consume 2 seg, le quedan 3.

[P2 | 2 -> 4] <- P2 consume 2 seg, le quedan 2.

[P1 | 4 -> 6] <- P1 consume 2 seg, le queda 1.

[P2 | 6 -> 8] <- P2 consume 2 seg, termina.

[P1 | 8 -> 9] <- P1 consume su último segundo, termina.

Este ejemplo demuestra cómo el simulador gestiona correctamente la alternancia (context switch) y la cola circular del algoritmo Round Robin.