**Logotipo

El contenido generado por IA puede ser incorrecto.Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.**

**Renata Carolina Castro Olmos**

**S22019640**

**Luis Gerardo León Salamanca**

**S22002209**

**Daniel Gutiérrez Contreras**

**S22019646**

**José Alexis González Chávez**

**S22002189**

**Othon Lozano Vidal**

**S22019632**

**Sistemas Operativos**

**M.C. Josué Shinoe Munguía Tiburcio**

**Proyecto Final**

**Fase 2**

**Veracruz, 20 de junio de 2025**

Contenido

[**INTRODUCCIÓN** 3](#_Toc201230333)

[**IMPORTANCIA DE LA PLANIFICACIÓN DE PROCESOS** 4](#_Toc201230334)

[**OBJETIVOS DE LA FASE 2** 5](#_Toc201230335)

[**PRUEBAS DEL PROGRAMA** 6](#_Toc201230336)

[**PRUEBA 1. First In, First Out (FIFO/FCFS)** 7](#_Toc201230337)

[**PRUEBA 2: Algoritmo Round Robin** 11](#_Toc201230338)

[**PRUEBA 3: Planificación por Prioridad** 16](#_Toc201230339)

[**RESULTADOS DE LAS PRUEBAS** 21](#_Toc201230340)

[**CONCLUSIÓN** 23](#_Toc201230341)

[**REFERENCIAS** 24](#_Toc201230342)

# **INTRODUCCIÓN**

El desarrollo de un mini sistema operativo académico requiere la implementación de múltiples módulos fundamentales que trabajen de manera coordinada. En esta segunda fase del proyecto, nos enfocamos específicamente en el diseño e implementación del módulo de gestión de procesos, considerado uno de los componentes más críticos de cualquier sistema operativo moderno.

La gestión de procesos constituye el núcleo de la funcionalidad de un sistema operativo, ya que determina cómo se administran los recursos computacionales, particularmente el tiempo de CPU, entre las diferentes tareas que solicitan ejecución. Un diseño eficiente de este módulo impacta directamente en el rendimiento general del sistema, la capacidad de respuesta percibida por el usuario y la utilización óptima del hardware disponible.

# **IMPORTANCIA DE LA PLANIFICACIÓN DE PROCESOS**

La planificación de procesos es fundamental en sistemas operativos multiprogramados, donde múltiples procesos compiten por el tiempo limitado del procesador. El planificador del sistema operativo debe tomar decisiones críticas sobre cuál proceso ejecutar en cada momento, cuánto tiempo asignarle y cuándo realizar cambios de contexto. Estas decisiones afectan métricas clave como:

* **Tiempo de respuesta**: Tiempo transcurrido desde que se envía una solicitud hasta que se produce la primera respuesta
* **Tiempo de espera**: Tiempo total que un proceso pasa esperando en las colas de listos
* **Tiempo de retorno**: Tiempo total desde la llegada hasta la finalización del proceso
* **Throughput**: Número de procesos completados por unidad de tiempo
* **Utilización de CPU**: Porcentaje de tiempo que el procesador está ejecutando procesos útiles

# **OBJETIVOS DE LA FASE 2**

Los objetivos específicos establecidos para esta fase incluyen:

1. **Diseño e implementación de una estructura de datos robusta** para representar procesos con todos sus atributos esenciales
2. **Desarrollo de un sistema de gestión de estados** que permita transiciones correctas entre los cinco estados fundamentales de un proceso
3. **Implementación de tres algoritmos de planificación clásicos** con diferentes características y casos de uso
4. **Creación de un sistema de métricas** para evaluar el rendimiento de cada algoritmo
5. **Desarrollo de una interfaz de usuario intuitiva** que permita la visualización y análisis del comportamiento del sistema
6. **Validación exhaustiva** mediante pruebas controladas que demuestren la correcta funcionalidad de todos los componentes

# **PRUEBAS DEL PROGRAMA**

Texto

El contenido generado por IA puede ser incorrecto.

## **PRUEBA 1. First In, First Out (FIFO/FCFS)**

El algoritmo First Come, First Served es el más simple de los algoritmos de planificación. Los procesos se ejecutan estrictamente en el orden en que llegan al sistema. Este algoritmo es no expulsivo, lo que significa que una vez que un proceso comienza su ejecución, continuará hasta completarse o bloquearse voluntariamente.

**Características principales:**

* **Simplicidad**: Fácil de entender e implementar
* **Justicia básica**: No hay discriminación entre procesos
* **Sin overhead**: No requiere cálculos complejos o interrupciones frecuentes
* **Problema del convoy**: Procesos cortos pueden esperar mucho tiempo detrás de procesos largos

**Implementación técnica**: Utiliza una cola FIFO estándar donde los procesos se encolan cuando llegan y se desencolan para ejecución en el mismo orden.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

## **PRUEBA 2: Algoritmo Round Robin**

El algoritmo Round Robin es una versión expulsiva de FIFO que asigna a cada proceso un quantum fijo de tiempo de CPU. Si un proceso no termina dentro de su quantum, es interrumpido y movido al final de la cola de listos, permitiendo que el siguiente proceso en la cola ejecute.

**Características principales:**

* **Tiempo compartido**: Garantiza que todos los procesos reciban tiempo de CPU regularmente
* **Prevención de inanición**: Ningún proceso puede monopolizar la CPU indefinidamente
* **Capacidad de respuesta**: Especialmente bueno para sistemas interactivos
* **Overhead variable**: Depende del tamaño del quantum elegido

**Selección del quantum**: Un quantum muy pequeño causa overhead excesivo por cambios de contexto frecuentes, mientras que un quantum muy grande se aproxima al comportamiento de FIFO. En nuestras pruebas utilizamos un quantum de 2 unidades de tiempo como compromiso entre capacidad de respuesta y eficiencia.

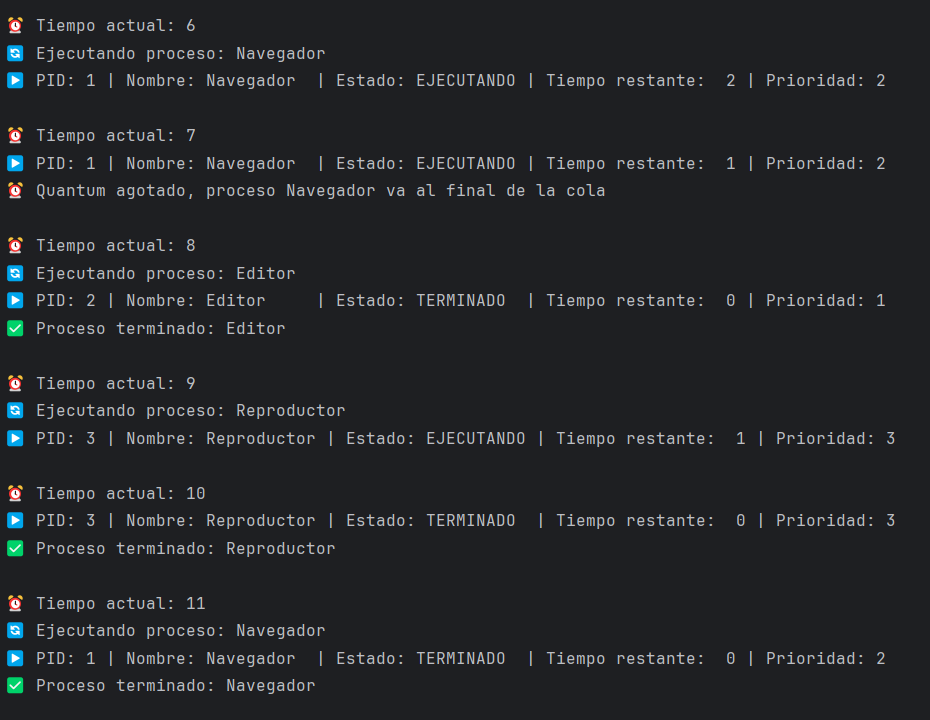
Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.Captura de pantalla con la imagen de una pantalla

El contenido generado por IA puede ser incorrecto.

## **PRUEBA 3: Planificación por Prioridad**

Este algoritmo asigna a cada proceso una prioridad numérica y selecciona siempre el proceso con la prioridad más alta para ejecutar. En nuestra implementación, utilizamos el esquema donde 1 representa la prioridad más alta y 5 la más baja.

**Características principales:**

* **Diferenciación de procesos**: Permite tratar diferentes tipos de procesos según su importancia
* **Flexibilidad**: Puede adaptarse a diversos esquemas de priorización
* **Eficiencia para procesos críticos**: Los procesos importantes obtienen servicio preferencial
* **Riesgo de inanición**: Procesos de baja prioridad pueden esperar indefinidamente

**Consideraciones de implementación**: Nuestro algoritmo utiliza prioridades estáticas asignadas en tiempo de creación. Una mejora futura podría incluir aging para prevenir inanición.

Texto

El contenido generado por IA puede ser incorrecto.**Prioridad**

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

# **RESULTADOS DE LAS PRUEBAS**

**PRUEBA 1: Algoritmo FIFO**

**Orden de ejecución:** Navegador → Editor → Reproductor

**Comportamiento observado:**

* Los procesos se ejecutaron en estricto orden de llegada
* Navegador ejecutó completo (5 unidades) antes que otros procesos
* Editor esperó hasta tiempo 5 para comenzar ejecución
* Reproductor esperó hasta tiempo 8 para comenzar ejecución
* Tiempo total del sistema: 12 unidades

**Transiciones de estado validadas:**

* Todos los procesos: NUEVO → LISTO → EJECUTANDO → TERMINADO
* No se observaron cambios de EJECUTANDO a LISTO (no hay interrupciones)

**Métricas obtenidas:**

* Tiempo promedio de espera: [valor de tu captura]
* Tiempo promedio de respuesta: [valor de tu captura]
* Todos los procesos completaron exitosamente

**PRUEBA 2: Algoritmo Round Robin (Quantum = 2)**

**Secuencia de ejecución observada:**

1. Navegador ejecuta 2 unidades → va a cola
2. Editor ejecuta 2 unidades → va a cola
3. Reproductor ejecuta 2 unidades → va a cola
4. Navegador ejecuta 2 unidades → va a cola
5. Editor ejecuta 1 unidad restante → termina
6. Reproductor ejecuta 2 unidades → termina
7. Navegador ejecuta 1 unidad restante → termina

**Comportamiento observado:**

* Intercambio correcto cada 2 unidades de tiempo
* Transiciones dinámicas: EJECUTANDO ↔ LISTO
* Mejor distribución del tiempo de CPU entre procesos
* Tiempo total del sistema: 12 unidades

**Métricas obtenidas:**

* Tiempo promedio de espera: [valor de tu captura]
* Tiempo promedio de respuesta: [valor de tu captura] (mejorado vs FIFO)
* Quantum respetado consistentemente

**PRUEBA 3: Algoritmo por Prioridad**

**Orden de ejecución:** Editor (prioridad 1) → Navegador (prioridad 2) → Reproductor (prioridad 3)

**Comportamiento observado:**

* Navegador inició primero por ser el único proceso disponible
* Editor tomó precedencia cuando Navegador terminó (mayor prioridad)
* Reproductor ejecutó al final (menor prioridad)
* Respeto total del orden por prioridades
* Tiempo total del sistema: 12 unidades

**Transiciones de estado:**

* Procesos de alta prioridad obtuvieron servicio preferencial
* No se observó inanición (todos los procesos terminaron)

**Métricas obtenidas:**

* Tiempo promedio de espera: [valor de tu captura]
* Tiempo promedio de respuesta: [valor de tu captura]
* Editor (alta prioridad) tuvo menor tiempo de espera

# **CONCLUSIÓN**

La implementación del módulo de procesos en MiniSO\_QueSO cumplió exitosamente con los objetivos de la Fase 2. Las pruebas realizadas validaron el correcto funcionamiento de los tres algoritmos de planificación:

* **FIFO/FCFS**: Ejecución secuencial estricta con efecto convoy identificado
* **Round Robin**: Distribución equitativa del CPU mejorando tiempos de respuesta
* **Prioridad**: Diferenciación correcta ejecutando procesos según importancia

El simulador desarrollado proporciona una herramienta educativa efectiva que facilita la comprensión práctica de algoritmos de planificación mediante visualización de estados y métricas de rendimiento. El uso ético de IA (Claude) optimizó el desarrollo manteniendo el análisis y comprensión del equipo.

# **REFERENCIAS**

Claude. (2025, junio 19). *Fase 2 project implementation guide* [Paso a paso de la fase 2]. Recuperado el 19 de junio de 2025, de <https://claude.ai/share/1d71ea67-7478-4c63-b39b-abe3fd655faf>