



# SISTEMAS OPERATIVOS

## **ACTIVIDAD DE APRENDIZAJE 6 PROGRAMA 3. ALGORITMO DE PLANIFICACIÓN FCFS (FIRST COME FIRST SERVER)**

**Profesor:**

**VIOLETA DEL ROCIO BECERRA VELAZQUEZ**

VERDUZCO ROSALES LUIS ENRIQUE

223992388

Ingeniería en Computación

28/09/25

CUCEI DIVTIC D04

## ÍNDICE

Objetivo de la actividad.....	2
Solución de problemas.....	2
Mayores problemas.....	3
Conclusión.....	3

---

## Objetivo de la actividad

Implementar un simulador de planificación de procesos basado en el algoritmo **FCFS (First Come, First Served)**, utilizando un modelo de cinco estados (Nuevo, Listo, Ejecución, Bloqueado y Terminado). El programa debe gestionar una memoria limitada, procesar interrupciones de usuario y calcular los tiempos de rendimiento para cada proceso.

## Solución de problemas

Para esta práctica se utilizó la combinación de **HTML, CSS y JavaScript**. Esta elección se debe a que permite crear simulaciones interactivas y visuales que se ejecutan directamente en cualquier navegador web moderno, sin necesidad de compilación o instalación de software adicional por parte del usuario. JavaScript es ideal para manejar la lógica del simulador, el control del tiempo y la manipulación del DOM, mientras que HTML y CSS permiten estructurar y dar estilo a la interfaz de forma clara y efectiva.

1. **Implementación del Modelo:** Se abandonó la estructura original de "lotes" para adoptar un modelo más preciso basado en los cinco estados. Se crearon arreglos (**arrays**) en JavaScript para gestionar cada cola:
  - **nuevos:** Almacena todos los procesos generados al inicio.
  - **listos:** La cola de procesos en memoria, listos para ser ejecutados.
  - **bloqueados:** Contiene los procesos que fueron interrumpidos por una operación de E/S (tecla 'E').
  - **terminados:** Guarda los procesos que han finalizado su ejecución, ya sea de forma normal o por error.
  - Una variable **procesoEnEjecucion** para el proceso que ocupa el CPU.
2. **Gestión de Memoria Limitada:** Se implementó la restricción de tener un máximo de 4 procesos en memoria a la vez (considerando los estados de Listo, Bloqueado y Ejecución). En cada ciclo del reloj del simulador (**tick**), se verifica si hay espacio disponible en memoria. Si es así y existen procesos en la cola de **nuevos**, se admite uno a la cola de **listos**, simulando su ingreso al sistema.
3. **Manejo de Interrupciones y Eventos:**
  - **Tecla 'E' (Interrupción):** El proceso en ejecución es movido a la cola de **bloqueados**. Se le asigna un contador de 8 segundos; una vez transcurrido ese tiempo, el proceso regresa a la cola de **listos**.
  - **Tecla 'W' (Error):** El proceso en ejecución se mueve directamente a la cola de **terminados** y su resultado se marca como "ERROR".

- **Teclas 'P' y 'C' (Pausa/Continuar):** Se implementó una variable booleana (`estaPausado`) que detiene o reanuda el avance del reloj global y la lógica del simulador.

## Mayores problemas

**Cálculo de Tiempos y Reporte Final:** A cada objeto de proceso se le agregaron propiedades para almacenar todas las medidas de tiempo requeridas.

- **Tiempo de Llegada:** Se registra cuando un proceso pasa de `nuevos` a `listos`.
- **Tiempo de Finalización:** Se registra cuando un proceso entra a la cola de `terminados`.
- **Tiempo de Servicio:** Se incrementa en cada `tick` que el proceso pasa en el estado de `ejecución`.
- **Tiempo de Espera:** Se incrementa en cada `tick` que el proceso pasa en la cola de `listos`.
- **Tiempo de Respuesta:** Se calcula como la diferencia entre el momento en que es atendido por primera vez y su tiempo de llegada.
- **Tiempo de Retorno:** Se calcula como la diferencia entre el tiempo de finalización y el de llegada.

Al finalizar la simulación, se genera una tabla de resumen con todos estos datos para cada proceso, lo cual creo que fue lo más complicado de lograr debido a todos los tiempos y la necesidad de verificar que fueran correctos.

## Conclusión

El algoritmo **FCFS** me pareció sencillo de implementar por ser secuencial, pero también pude ver su principal desventaja: si un proceso largo llega primero, puede hacer que procesos más cortos esperen innecesariamente.

Finalmente, el cálculo de los tiempos me ayudó a cuantificar la eficiencia del planificador y a entender cómo se mide el desempeño de un sistema operativo en términos de la gestión de sus recursos más críticos, como el tiempo de CPU.

**Código Fuente:** <https://github.com/UnluckyLuren/SisOperativos.git>

**Ejecutable:** <https://unluckyluren.github.io/SisOperativos/>