SISTEMAS OPERATIVOS

ACTIVIDAD DE APRENDIZAJE 8
PROGRAMA 4

Profesor:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

12/10/25

CUCEI DIVTIC D04

ÍNDICE

Objetivo de la actividad

En esta práctica se desarrolló un programa para simular la planificación de procesos utilizando el algoritmo FCFS (First Come, First Served). El programa administra los diferentes estados de un proceso (nuevo, listo, en ejecución, bloqueado y terminado) y gestiona las transiciones entre estos estados. Además, se hizo interactiva la simulación, permitiendo al usuario realizar acciones específicas como crear nuevos procesos, interrumpirlos por eventos de entrada/salida, finalizarlos por error o pausar la ejecución del sistema a través de teclas específicas.

Lenguaje Utilizado

Para la realización de esta práctica se utilizó JavaScript, junto con HTML para la estructura y CSS para los estilos. Elegí JavaScript porque permite crear un entorno más visual e interactivo que se ejecuta directamente en un navegador web, facilitando la visualización en tiempo real de cómo cambian de estado los procesos y las colas.

Soluciones Implementadas

La lógica principal del código se basa en un ciclo que se actualiza cada segundo para simular el paso del tiempo.

1. Estructura del Proceso:
   Para cada proceso, creé un objeto de JavaScript. Este objeto funciona como el Bloque de Control de Proceso (BCP) y guarda toda su información: un id único, la operación matemática como texto ("5 * 8"), su Tiempo Máximo Estimado (tme) generado aleatoriamente, y variables para todos los tiempos importantes: tiempoLlegada, tiempoFinalizacion, tiempoRetorno, tiempoServicio, etc.

2. Estados y Colas: Utilicé arreglos (arrays) de JavaScript para manejar las diferentes colas y estados:

   - nuevos: Almacena los procesos recién creados.
   - listos: Funciona como la cola de listos, donde los procesos esperan para usar el CPU.
   - bloqueados: Contiene los procesos interrumpidos por una solicitud de E/S.
   - terminados: Guarda los procesos que ya finalizaron.
   - procesoEnEjecucion: Una variable que contiene el objeto del proceso que está usando el CPU en ese momento.

3. Ciclo Principal y Reloj Global:
   La función más importante es mainEjecucion, que se ejecuta repetidamente gracias a un setInterval(mainEjecucion, 1000). Esto hace que el relojGlobal del sistema avance un segundo en tiempo real. En cada ciclo (cada segundo), el programa:

- Revisa los procesos bloqueados:
  Si un proceso ha cumplido su tiempo de bloqueo (8 segundos), es movido a la cola de listos.

- Admite nuevos procesos:
  Si hay espacio en memoria (un máximo de 4 procesos entre listos, bloqueados y en ejecución), el planificador a largo plazo mueve un proceso del arreglo de nuevos a la cola de listos.

- Procesa el trabajo:
  Avanza el tiempoServicio del proceso en ejecución y verifica si ya terminó su TME.

- Despacha un nuevo proceso:
  Si el CPU está libre, se aplica la lógica FCFS tomando el primer proceso de la cola de listos con el método listos.shift().

4. Implementación de las Teclas de Control: Usé un addEventListener para obtener las pulsaciones de teclas. Un switch se encarga de ejecutar la acción correspondiente:

   - 'N' (Nuevo):
     Llama a una función para crear un nuevo objeto de proceso y lo agrega al arreglo nuevos.

   - 'E' (Interrupción):
     Mueve el procesoEnEjecucion al arreglo de bloqueados y deja el CPU libre.

   - 'W' (Error):
     Llama a la función terminarProceso(true), pasándole un true para indicar que la terminación fue por error. El resultado se marca como "ERROR".

   - 'P' y 'C' (Pausa/Continuar):
     Simplemente cambian el valor de una variable booleana estaPausado. El ciclo principal no se ejecuta si esta variable es true.

   - 'B' (Tabla BCP):
     Pausa la simulación y llama a una función que reúne los procesos de todos los arreglos (nuevos, listos, etc.), los ordena por ID y construye dinámicamente la tabla BCP en una ventana modal, mostrando el estado y los tiempos de cada uno.

5. Cálculos y Visualización:
   Para calcular el resultado de las operaciones, utilicé la función eval() de JavaScript sobre el texto de la operación (eval("5 * 8")). Los tiempos de cada proceso se calculan y actualizan según las fórmulas. Toda la interfaz de usuario se actualiza en cada ciclo para reflejar en tiempo real el contenido de las colas, el estado del proceso en ejecución y el reloj global. Al finalizar la simulación, se muestra una tabla final con todos los datos de los procesos terminados.

Conclusión

Esta práctica me ayudó a entender de una forma muy visual y clara la dinámica de la planificación de procesos. Usar JavaScript fue de gran ayuda para mi por todas sus herramientas y para mejorar en la programación web.
El algoritmo FCFS es muy fácil de implementar, ya que se comporta como una fila normal. La parte más importante para mi fue programar las interrupciones y ver cómo un proceso puede ser pausado y luego reanudar su ejecución en el sistema. El mayor desafío fue organizar mi código para actualizar correctamente todos los tiempos y la pantalla en cada segundo sin que nada fallara.

Código Fuente: https://github.com/UnluckyLuren/SisOperativos.git

Ejecutable: https://unluckyluren.github.io/SisOperativos/