SISTEMAS OPERATIVOS

ACTIVIDAD DE APRENDIZAJE 2
PROGRAMA 1

Profesor:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

31/08/25

CUCEI DIVTIC D04

ÍNDICE

Objetivo

Simular el Procesamiento por Lotes.

Notas:

Elegí Javascript (js) por su versatilidad a la hora de manejar objetos, ya que en js existen los objetos literales los cuales te permiten guardar varias llaves y datos, lo cual me sirvio de mucho para manejar de mejor manera los datos que me daba el usuario para cada proceso.

HTML y CSS complementaron a JS al permitirme tener un entorno más fácil de configurar para pedir los datos y luego realizar los cálculos correspondientes para cada lote.

Es un sistema que:

- Crea procesos con operaciones matemáticas
- Los agrupa en lotes de 4 procesos cada uno
- Ejecuta cada lote simulando el tiempo que tardaría
- Muestra resultados en tiempo real con cronómetros
- Lleva un contador global de todo el proceso

Variables Principales

```
const arrObjetos = [];  // Aquí guardo todos los procesos que se van creando
```

Flujo del Programa

1. Crear procesos - con nombre, números, operación y tiempo aleatorio
2. Formar lotes - grupos de 4 procesos cada uno
3. Ejecutar por lotes - mostrando en pantalla con tiempos reales
4. Mostrar resultados - en tablas al finalizar

Funciones más importantes

```
crearObjInfo() - Crea objetos proceso:
{
   nombre: "Proceso1",
   numA: 5,
   numB: 3,
   opr: "+",          // Operación
```

```
    tiempo: 6,          // Tiempo aleatorio entre 1-10
    id: 1               // ID único
}
```

crearLotesSeparados() - Divide los procesos en grupos de 4:

```
        // Si tengo 7 procesos, crea:
        lote1 = [proceso1, proceso2, proceso3, proceso4]
        lote2 = [proceso5, proceso6, proceso7]
```

imprimirCiclicamente()

- Muestra los cronómetros con tiempo el transcurrido y el restante
- Cambia los colores cuando está por terminar (se pone rojo)
- Actualiza en tiempo real cada 100ms

Partes Complicadas

Los async/await:

```
async function imprimirCiclicamente() {
    await new Promise(resolve => setTimeout(resolve, intervalo));
    // Esto hace que espere exactamente el tiempo del proceso
}
```

Dificultad Técnica, por:

- Programación asíncrona
- Manipulación del DOM
- Gestión de tiempos múltiples
- Algoritmos de agrupamiento

Conclusión

Organicé el código en funciones modulares que se encargan de tareas específicas: creación de objetos, formación de lotes, ejecución cíclica y cálculo de resultados. Esta modularidad hace que el código sea mantenible y escalable.

El simulador no solo funciona técnicamente bien, sino que me proporcionó una experiencia valiosa para entender cómo se procesaban los trabajos en lotes en los

sistemas operativos de antes. Los efectos visuales de los cronómetros y la transición entre lotes crean una representación auténtica del proceso.

Referencias

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

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