Universidad de Guadalajara

CUCEI (Ciencias Exactas e Ingenierías)

Departamento de ciencias computacionales

Materia: Sistemas operativos

Profesor: Violeta del Rocío Becerra Velazquez

Gómez Rubio Alexia

Rico Morones Denice Estefania

Carrera: Ingeniería en computación

Sección: D04

Programa 4

Tema: Algoritmo de planificación (FCFS)

Fecha: 13 Octubre 2024

Índice

Objetivo

Crear un programa, en el que se registren “n” procesos,, en cada proceso se generarán los datos aleatorios para el nombre, ID, operación a realizar (suma, resta, multiplicación, división, módulo), el tiempo máximo estimado que el programa tardará en ejecutar esa operación.

Este programa se controlara con algunas teclas, con la letra “e” terminamos el programa forzandolo como un error, con la letra “i” se mandan directamente a bloqueados y se quedan ahí un tiempo de siete segundos, con la letra “p” se detiene la ejecución del programa momentáneamente, hasta que se presione la letra “c”, la cual reanudará el programa que se pauso, la tecla “n” crea nuevos procesos con datos aleatorios y la tecla “t” muestra la tabla de los procesos con su información.

Se están contemplando el diagrama de los cinco estados, en la ejecución del programa: nuevos, listos, ejecución, bloqueados y terminados. Los procesos se van a ejecutar en el orden en que son ingresados, a menos de que sean finalizados por error, o se manden a bloqueados. Con ayuda de un contador global, se sumarán los tiempos estimados de todos los procesos registrados hasta terminar todos.

Se están registrando los tiempos de llegada (hora en que el proceso entra al sistema), finalización (proceso finaliza su ejecución), retorno (tiempo total desde que el proceso llega hasta que termina), respuesta (tiempo transcurrido desde que llega en que es atendido), espera (tiempo que ha estado esperando para usar el procesador) y servicio(el tiempo que el proceso ha estado en el procesador).

Al final de que todos los procesos hayan sido ejecutados, se mostrará toda la información de los procesos, tanto lo que se generó aleatoriamente como los tiempos.

Descripción del programa

Lenguaje utilizado:

Decidimos seguir utilizando Javascript y HTML para facilitar la demostración gráfica del programa, y porque solo le cambiamos/agregamos al programa anterior unas funciones que nos hacían falta.

Una de las ventajas del uso de Javascript junto con HTML para mostrar los resultados gráficamente dentro de nuestro servidor local es que a comparación de un lenguaje que se maneja en consola, es mucho más fácil de manipular y utilizar para acomodar gráficamente los datos de una manera muy sencilla sin la necesidad de recurrir a librerías para el manejo de las coordenadas en consola; no se está utilizando ningún framework.

Estructura del programa:

Al igual que en anterior dividimos el código HTML y el código de JavaScript para tener una mejor organización en nuestro programa.

En la parte de html nos apoyamos de las etiquetas para representar gráficamente la captura de los procesos como la ejecución de cada uno de ellos, además, también nos ayuda a ejecutar y mandar llamar a la función que dentro del .js nos da la funcionalidad del programa, esto, gracias a los <form>.

Para la parte del JavaScript agregamos al inicio del código declaramos alguna variables globales, como: procesos el cual almacena los procesos generados, lotes guarda los procesos en grupos de hasta cinco procesos, loteEnEjecucion contiene el lote actual que está siendo procesado, el procesoActual guarda el proceso específico que se está ejecutando, contadorGlobal lleva la cuenta del tiempo total desde el inicio de la ejecución, intervaloGlobal es el temporizador que incrementa el contadorGlobal cada segundo, intervaloProceso controla la ejecución de cada proceso, idContador es un contador que asigna un identificador único a cada proceso, isPaused es una bandera que indica si la ejecución está pausada o no.

El código incluye un mecanismo para pausar y continuar la ejecución de los procesos. Cuando el usuario presiona la tecla "p", la función pausar() detiene ambos intervalos de tiempo y con una alerta se le informa al usuario de su estado. Si el usuario presiona "c", la función continuar() reanuda los intervalos y retoma la ejecución del proceso actual.. La función Procesos() crea una cantidad de procesos

basada en la entrada del usuario, asignando nombres, operaciones y tiempos aleatorios. Estos procesos se agrupan en lotes mediante la función actualizarLotes(), que mueve los procesos a lotes y actualiza el contador de lotes pendientes.

Para iniciar la simulación, la función Iniciar() se encarga de arrancar el contador global y llamar a ejecutar(), que gestiona la ejecución de los lotes. La función ejecutar() toma el primer lote de la lista de lotes y lo coloca en ejecución. A través de la función mostrarLote(), se actualiza la interfaz visual para mostrar los procesos del lote en ejecución. Luego, la función ejecutarProceso() comienza a procesar los elementos del lote uno a uno, decrementando su tiempo restante en intervalos de un segundo. Si el tiempo de un proceso llega a cero, se calcula el resultado de su operación y se mueve al siguiente proceso o lote.

La función de handleInterruption() se activa cuando el usuario presiona la tecla "i". Manda en automático el proceso a bloqueados, y se espera ahí un tiempo de siete segundos y regresa a listos para continuar con su ejecución.

De igual forma la funcion handleError() se activa cuando el usuario presiona la tecla "e". El proceso actual es finalizado repentinamente y se registra en la interfaz como "terminado con error". Luego, se pasa al siguiente proceso del lote o, si el lote ha terminado, se continúa con el siguiente lote. Esta función representa una falla crítica en la ejecución de un proceso, que no puede completarse de forma correcta.

El procesamiento de cada operación matemática es realizado por la función Operacion(), que ejecuta la operación correspondiente al tipo de proceso, como suma, resta, multiplicación, división o módulo. Una vez que un proceso finaliza su ejecución, el resultado es mostrado en la interfaz mediante la función procesarResultado(), la cual asegura que no se dupliquen los resultados y que los procesos ya terminados se registren correctamente. El ciclo continúa hasta que todos los lotes han sido procesados y la simulación concluye, algo importante de mencionar es la modificación de esta función procesarResultado() ya que se nos pide mostrar la tabla relacionada que representa al BCP, la cual se hace mediante inserciones para las filas y columnas de la misma, esto gracias a la interacción que se puede realizar mediante el DOM; la creación de más atributos para los procesos

nos fueron de gran ayuda ya que estas variables como tiempoDeLlegada, tiempoDeInicio y tiempoDeFinalizacion son pieza clave para después en la función procesarResultado() podamos hacer las operaciones para calcular el tiempo de servicio, el tiempo de retorno y el tiempo de espera.

Además, se agregaron funciones para al momento de la interrupción poder mandar el proceso al nuevo apartado “bloqueados” el cual se realiza mediante las funciones bloquearProceso() y mostrarbloq() donde la primera se encarga de verificar si hay más procesos pendientes en el lote de ejecución para así bloquearlo y pasar al siguiente proceso; y la segunda, es una función encargada de interactuar con el DOM para poder mostrar el listado de los procesos que han sido bloqueados.

La función tabla() verifica si se está mostrando la tabla o no, cuando se presiona la tecla “t”, si la tabla no está visible la muestra, pero si ya está visible la oculta.

Finalmente en este programa se agrego el evento para la tecla “n” el cual agrega un nuevo proceso al presionar esta tecla mientras el programa se ejecuta, esto haciendo uso de una nueva función llamada nuevoProceso la cual es parecida a la función inicialmente creada llamada Procesos, la diferencia, es que en esta lo que hacemos es verificar el largo de la memoria de los procesos en ejecución para que asi si aun hay espacio para un proceso pueda entrar directamente a Listos si no, se va Nuevos.

Problemas presentados y su solución:

Algunos de los problemas presentados fueron a la hora de añadir los nuevos procesos ya que nos teníamos que asegurar de que entraran directamente a Listos, lo cual al probar la función por primera vez no resultaba de esa manera por lo que tuvimos que hacer algunas modificaciones y añadir condicionales para poder añadirlos donde van según las condiciones que se tienen.

Conclusiones

Gómez Rubio Alexia

Para este programa no tuvimos muchas complicaciones, al tercer programa solo le agregamos unas funciones para complementarlo (tecla n y t), y solo terminar de ajustar los tiempos, porque algunos no se registraban bien, en comparación con programas anteriores este fue uno de los que menos problemas hemos tenido.

Rico Morones Denice Estefania

Este programa me pareció uno de los más sencillos ya que fueron pocas las modificación que le tuvimos que realizar por lo que no fueron muchos problemas a los que nos enfrentamos en comparación a programas anteriores, considero que esta práctica fue todo un éxito y cumple con lo que se solicita.