ABARCA VIQUEZ ANDRES

INFORME DEL PROYECTO

15 de agosto de 2017

emulador de Planificador de disco secundario a nivel de sistema operativo

Chavy

# Solución planteada

Para este proyecto se optó por hacer uso del lenguaje orientado a objetos C#, usando el framework de Microsoft .NET, para así lograr un solución con un tiempo de respuesta alto, fácil acceso por medio de Windows Forms.

La solución consta de dos clases: objeto disco de almacenamiento secundario con los atributos de la cola de peticiones de acceso a los cilindros del disco, el tamaño del disco y la posición del cabezal para emular la lectura. La segunda clase contiene todas las funciones necesarias para obtener los datos del objeto anteriormente definido y las funciones para ordenar los valores para cada algoritmo de planificación y finalmente, las funciones para graficar el desplazamiento por algoritmo.

La solución planteada hace uso de algoritmos creados empíricamente para la planificación variada, FIFO, SSF, SCAN, CSCAN, y se grafican mediante la clase “Charts” del framework.

# Resultados obtenidos

La solución planteada cumple los objetivos del problema, por medio de la clase chart, se insertó en un Windows Form, tres objetos (text box) para la inserción y obtención de los datos del disco, en donde se obtiene el tamaño del disco, la cola de peticiones de acceso al disco y el punto de inicio. La solución además cuenta con un objeto tipo chart para la graficación de los movimientos de la planificación de los distintos algoritmos.

Para la graficacion de los algoritmos de planificación, se planteó una organización de los valores de la cola de peticiones. Por ejemplo, para el algoritmo SSJ, se creó un nuevo arreglo de los valores de la cola de peticiones que son ordenados por medio de la sintaxis lambda de c#, los valores más cercanos al punto de partida. Para el algoritmo FIFO, no se cambió el orden de los valores de la cola de peticiones. Para el algoritmo SCAN, se implementó una función que ordena los valores menores al punto de inicio de manera descendente hasta llegar a cero, y los valores mayores al punto de inicio de manera ascendente hasta llegar al último punto de la cola de peticiones. Para el algoritmo CSCAN, se implementó una función que ordena los valores mayores al punto de inicio de manera ascendente hasta llegar al valor mayor del disco, y después brincar al punto cero, después se ordenaron los valores menores al punto de inicio de manera ascendente hasta llegar al último punto de la cola de peticiones restante.

Todos los algoritmos fueron graficados en el mismo chart del Windows form.

# Benchmarking