UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e Ingenierías (CUCEI)

Departamento de Ciencias Computacionales

Sistemas Operativos

M. en C. Violeta del Rocío Becerra Velázquez

Garcia Saldivar Hugo Gabriel

220530758

Ingeniería en Computación (ICOM)

D05

Actividad 10 Programa 5. Algoritmo de Planificación RR (Round-Robin)

26 de octubre del 2025

Programa 5. Algoritmo de Planificación RR (Round-Robin).

Contenido

Tabla de imágenes

Contenido

Modificaciones en el main y funciones

La modificación principal para este programa 10 modificar nuestro algoritmo de planificación FCFS (First-Come First-Served) a uno RR (Round-Robin), este algoritmo de planificación se basa en darle a cada algoritmo de ráfagas de tiempo en la CPU, esto determinado por un valor denominado Quantum que no es mas que el responsable de definir el tiempo máximo que pasará un proceso en la CPU (es decir, el tiempo de la ráfaga). Por lo que en nuestra función Main añadiremos y modificaremos el bloque try-catch para poder recibir un valor entero mayor a 0 destinado al Quantum para el programa.

Una vez que recibimos y filtramos correctamente el valor para el Quantum, es momento de adaptarlo para la lógica de nuestro programa, la función que modificaremos será principalmente la *simulateProcessing*; Esta función ahora recibirá el valor que se defina para el Quantum, con el fin de poder simular correctamente el RR.

void simulateProcessing(vector<Process>& newProcesses, int numberProcesses, vector<unsigned int>& usedIds, int quantumValue);

Contador para el Quantum

Para mantener un control en la función *simulateProcessing* vamos a añadir un contador del quantum, esto para que el valor no exceda el limita marcado y el proceso no este mas tiempo del debido en la CPU.

int quantumCounter = 0;

Ahora aplicaremos una lógica de expulsión, esto para que cuando el quantum llegue a su valor máximo y el proceso aun no haya terminado (o, por otro lado, el quantum aun no termine y el proceso finalice) el proceso sea enviado a la cola que le corresponda.

Por lo tanto, seguiremos este flujo:

1.- Si el proceso finalizo normalmente, saldrá a la cola de procesos finalizados.

2.- Si el proceso no ha finalizado, pero se cumple la condición *quantumCounter* >= *quantumValue* el proceso será expulsado a la cola procesos listos (creando una copia y eliminando el proceso para simularlo).

En cualquiera de los 2 casos, es importante que el contador *quantumCounter* se reinicie nuevamente a 0 para que, si existe un proceso en espera, pueda entrar y procesarse respetándose su ráfaga de tiempo. A su vez, el contador se reiniciará si se presiona alguna tecla especial *_kbhit* que afecte al proceso actual en ejecución como “W” o “E”.

Programa 5. Algoritmo de Planificación RR (Round-Robin).

Modificación de consola

Como todas las preparaciones, modificaremos la función *displayStatus* para que el usuario pueda ver el valor del Quantum en tiempo real, decidimos hacerlo añadiendo un nuevo atributo en los procesos en ejecución llamado "Quantum Actual" que utiliza la linea *to_string(quantumCounter)* para mostrar dinámicamente el quantum que se va teniendo hasta el momento, cuando el Quantum deba reiniciarse a 0 el mismo se vera reflejado en el programa.

En la Figura 1 se muestra una captura de pantalla general de la consola:

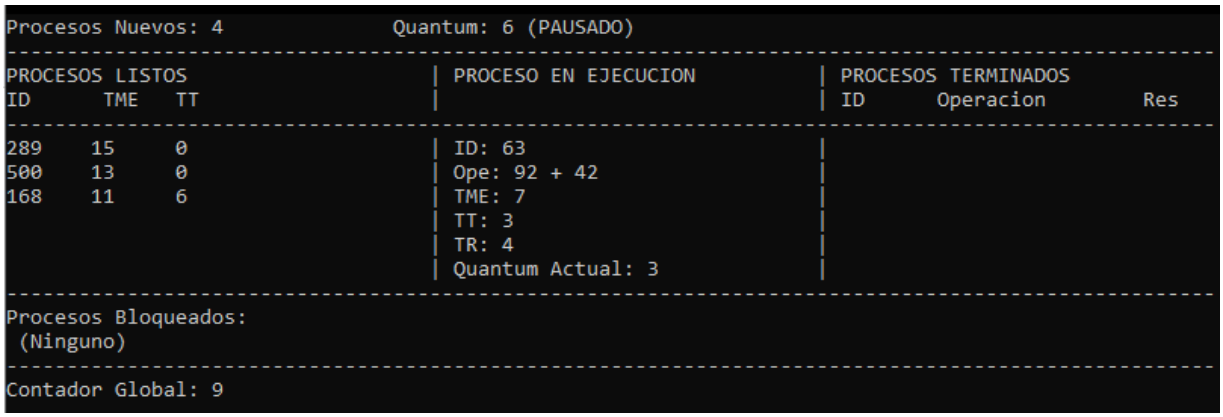

Figura 1. Captura general de la consola.

Para esta captura se ingresaron 8 procesos y se estableció un Quantum de 6, se puede ver claramente como el proceso ID 168 ya habia pasado por la CPU al ser el primer proceso entrante, despues de que su ráfaga de tiempo pasara (aun sin ser finalizado) el mismo se formo al final de la cola de procesos listos y el proceso ID 63 entro a tomar su ráfaga de tiempo.

En la Figura 2. Se puede comprobar que el proceso ID 168 fue el primer proceso entrante como se describió anteriormente, esto mediante la tabla BCP final.

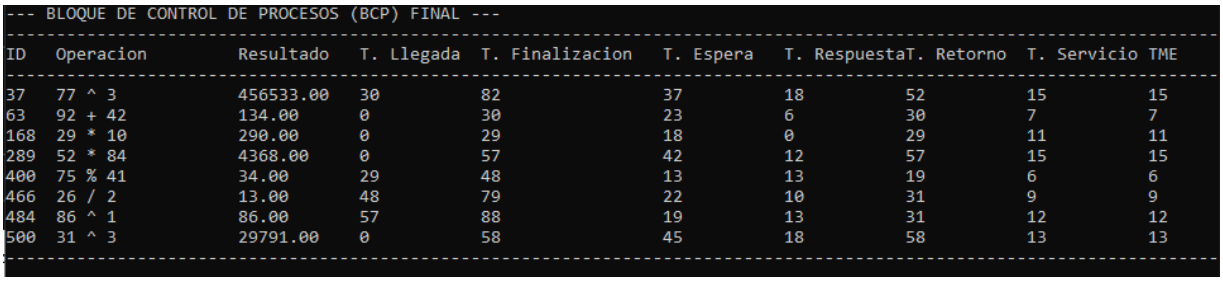

Figura 2. Captura final tabla BCP.

Conclusiones

Honestamente fue muy sencilla la lógica y creación de este nuevo programa, principalmente creemos que fue de esta forma debido a nuestros conocimientos previos de los algoritmos de planificación gracias a la realización a actividad anterior donde estos algoritmos se vieron relacionados, realmente entender desde un punto de vista mas práctico el algoritmo de planificación RR causo que inmediatamente nos diéramos una idea muy rápida de como lograr resolver este problema.

El tener un buen entendimiento previo de este algoritmo fue la clave para que el programa fuera exitosamente completado, además de que fue la única solicitud para esta entrega.