****

**GARBAGE COLLECTOR EN LENGUAJE C**

Gonzalo Bravo

Eduardo Aguilar

Laura Velazquez

Mariela Carriqueo

UNIVERSIDAD NACIONAL DE RÍO NEGRO

PROGRAMACIÓN I

Miguel Mariguin

Daniel Teira

14 de junio de 2023

Un garbage collector es un agente que administra un bloque de memoria para un conjunto de programas que necesitan ejecutar sus algoritmos. El objetivo es administrar la memoria de cada bloque y una vez que el cliente termina de utilizarlo, borrarlo y liberar la memoria.

Para la implementación de este programa en C se desarrollaron 9 funciones para que el sistema administre dinámicamente la memoria en función a los requerimientos del cliente.

A tal fin se utilizaron las siguientes librerías:

* Stdio.h
* Stdlib.h
* String.h
* Garbage.h: donde se realiza la declaración de interfaz de funciones

El programa se implementó mediante la ejecución de múltiples arreglos que permiten la administración de los bloques de memoria, su tamaño, las referencias y la cantidad utilizada y por ende disponible de memoria.

Las funciones implementadas son:

1. int init gc(int maxmem): en esa función se declaran las variables del módulo y se inicializan los distintos arreglos asignándole a cada uno el espacio de memoria dinámica reservada. La función da inicio al garbage collector declarando la memoria máxima y generando un error si la misma es menor o igual a 0. En esta instancia el sistema invita al cliente a ingresar la cantidad de memoria que se necesita para un nuevo bloque y el texto con la firma del proceso.
2. int new\_block(int sz, char \*name): se asigna la memoria dinámica al bloque y al arreglo de referencias así como el tamaño al arreglo de tamaños en la creación de los nuevos bloques de memoria. Se asigna la cantidad de referencias en el arreglo correspondiente así como el nombre del bloque.
3. int \*mem\_ptr(int block): Se asigna un puntero a la dirección de memoria reservada.
4. int resize(int block, int sz): La función permite ampliar la memoria asignada a un determinado bloque.
5. int add\_reference(int block): La función asigna un nombre al nuevo bloque de memoria incrementando el contador de referencias del bloque dado por el arreglo block.
6. int remove\_reference(int block): En esta función cada que vez que la variable termina su tiempo de vida, el compilador le da aviso al garbage collector que decrementará el contador de referencias. Cuando el contador de referencias se iguala a 0, se liberan todas las referencias y se debe liberar la memoria de los arreglos de referencias y punteros.
7. int cur\_used\_memory(void): Se suma iterativamente el tamaño de todos los bloques reservados para determinar el total de memoria en uso.
8. int cur\_available\_memory(void): Calcula la cantidad de memoria disponible en función a la memoria máxima y la cantidad en uso. Si hay memoria disponible, el sistema reserva la porción de memoria informando por pantalla el número de bloque, la cantidad de memoria reservada y la identificación. Luego consulta al cliente si desea reservar un nuevo bloque de memoria. Si la cantidad memoria disponible es insuficiente para reservar un nuevo bloque, el sistema informa por pantalla que no es posible reservar la cantidad deseada y cuánto es la memoria disponible, invitando al cliente a reservar una cantidad menor para el bloque deseado. Cuando la cantidad de memoria disponible es 0, el sistema informa que ya no hay memoria disponible.
9. int destroy\_agent(): Una vez que finaliza la vida del sistema, el garbage collector libera toda la memoria utilizada hasta el momento.