Project Report

On

GARBAGE COLLECTION

In the fulfilment of

6th semester of

ICT

By
Darpan Patel (121008)
Shreya Gokani (121053)

Faculty:-

Dr. Hemal Shah

IET
Igniting Innovation
INSTITUTE OF ENGINEERING AND TECHNOLOGY

AHMEDABAD UNIVERSITY
Global education at local cost, context and ethos

COURSE OF OPERATING SYSTEM

INSTITUTE OF ENGINERRING AND TECHNOLOGY

Contents

ACKNOWLEDGEMENTS

We have a great pleasure in acknowledging the help given by various individuals throughout the project work. This project is itself an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals.

We express our sincere and heartfelt gratitude to Dr. Hemal Shah, Professor of the course Operating System, for being helpful and co-operative during the period of the project and for their valuable guidance, timely suggestions and help in the completion of this project.

We extend my sincere thanks to Teaching Associates and all the non-teaching staff as well as to those who have helped us directly or indirectly for providing the necessary facilities and help. Without the support of anyone of them this project would not have been reached at this stage.

SYNOPSIS

OBJECTIVE:-

The Central objective of Garbage Collection is to provide basic facility of removing unused objects from memory for programmer in C language.

DURATION:-

2 months

TOOLS USED FOR DEVELOPING:-

Language used -> C, Java

Compiler used -> gcc

Libraries & tools used ->

- Linux Terminal
- gedit
- signal.h
- unistd.h
- stdlib.h
- stddef.h
- stdio.h
- setjmp.h

1. PROJECT IDEA AND APPLICATION: -

Garbage collection frees the developer from tracking memory usage and knowing when to free memory. If unreferenced locations are not collected, that memory is count wasted and may lead to unavailability of memory even if there exists available memory which cannot be used during the execution of the program. Garbage collection is provided in many languages like java, c# nowadays.

In our project we have tried to develop a library that collects garbage objects, into the 'C' programming language. In our library we write routines that allocates memory dynamically in to the 'C'programming language. Basically we modified malloc function. Library developed works as programmer's interface to our garbage collector. The interface is simple and easy like we include library in starting of the program.

2. INTRODUCTION:-

Garbage collection is a part of a language's runtime system, or an add-on library assisted by the compiler that automatically determines what memory a program is no longer using and recycles it for other use. It is also known as "automatic storage (or memory) reclamation". In the collector that we have designed, it is add-on library and is not assisted by the compiler / OS.

3. DETAILED DESCRIPTION:-

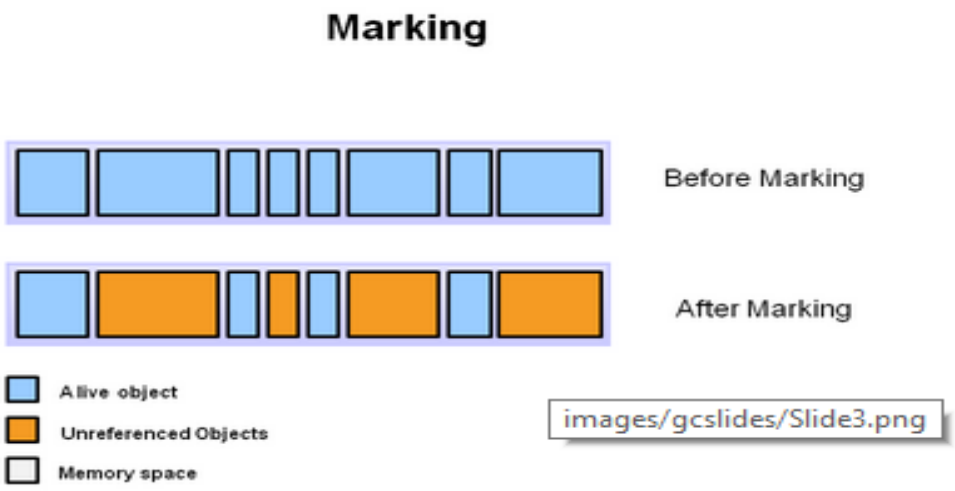
C is one of the most used programming languages in the world. We have done a lot of programming in C used the built in functions like "malloc", "calloc" etc. defined in the standard library to allocate memory and when the allocated memory is not needed any more we use the built in function "free" to free up the allocated memory. We have tried to improve the functionality of the 'C' programming language by developing a library that supports garbage collection (frees up the memory when required). Garbage Collection feature is available in language like JAVA. Kernel level modification is under research. So we have implemented garbage collection at user level in 'C' language. Our library keeps track of memory allocation & marks it to check whether object

allocated is used or has become garbage. If any programmer uses our library for allocating memory dynamically, we frees up the allocated memory.

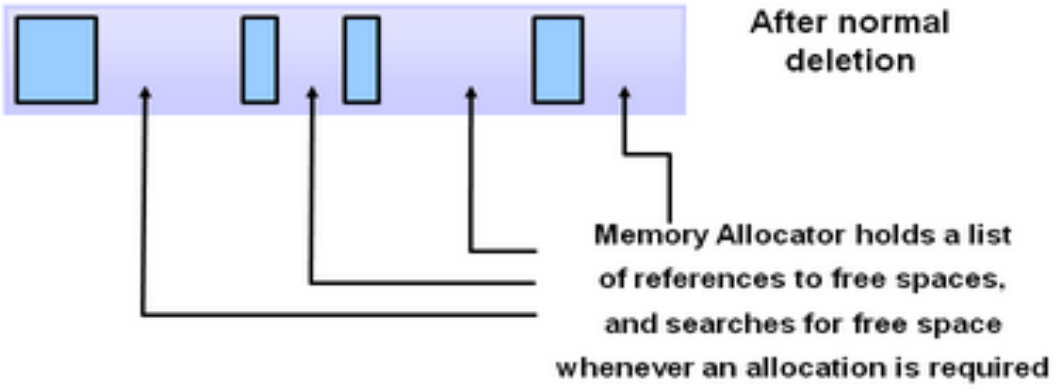
We have implemented garbage collection using conservative method i.e., mark and sweep algorithm.

4. MARK AND SWEEP ALGORITHM:-

The mark-and-sweep algorithm consists of two phases: In the first phase, it finds and marks all accessible objects. The first phase is called the mark phase. In the second phase, the garbage collection algorithm scans through the heap and reclaims all the unmarked objects. The second phase is called the sweep phase.



Normal Deletion

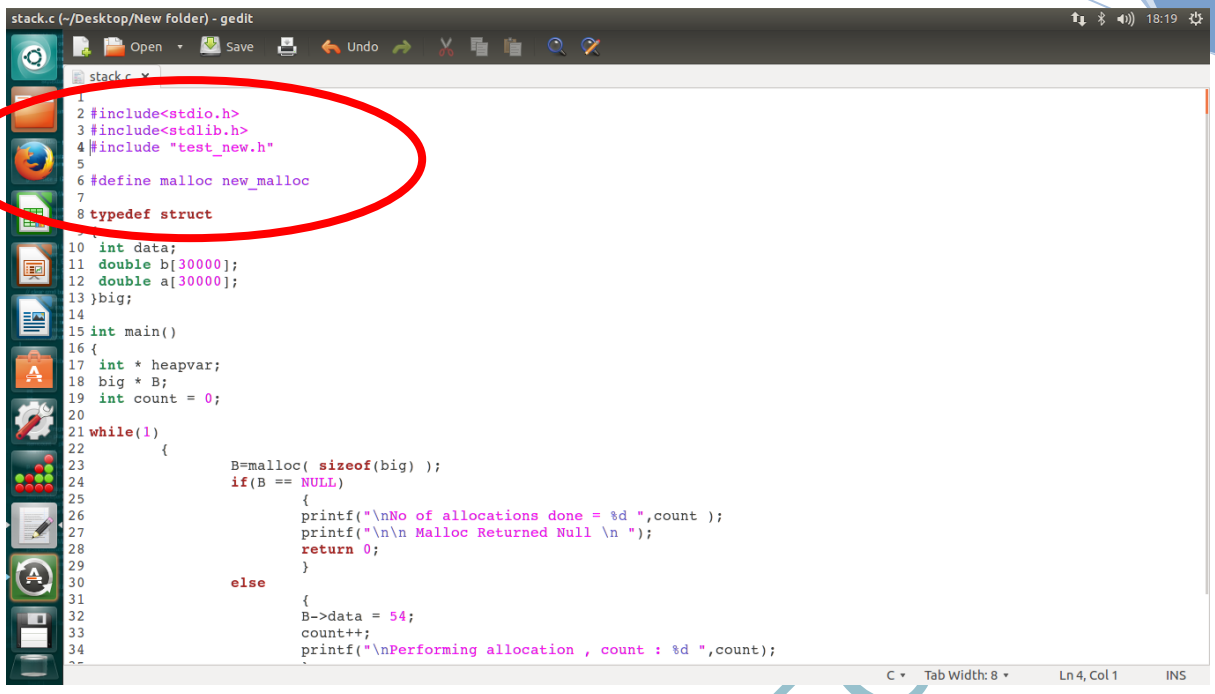


1. The user requests for allocation using the function “GC_malloc” or any of its variants.
2. We try to satisfy his request by allocation an object of the size requested by the user from the heap.
3. If the allocation in step 2 succeeds then we return the address of the allocated object after doing record-keeping activity
4. If during the process of allocation in step 2 we were unable to allocate or if the allocation so far done has exceeded a threshold limit (640kb by default), then we call our Garbage Collection routine that tries to collect unused memory (i.e. garbage).
5. Our GC Routine works as follows:
   From the addresses that have been allocated (these are available from the record keeping entries) search which of the objects pointed to by these addresses are live and which are not. This can be found by
   a. First making a search that will tell us help us to find the set of root pointers.
   b. The record keeping list gives us the information about what to search? And the root pointers will give us the information about where to search?

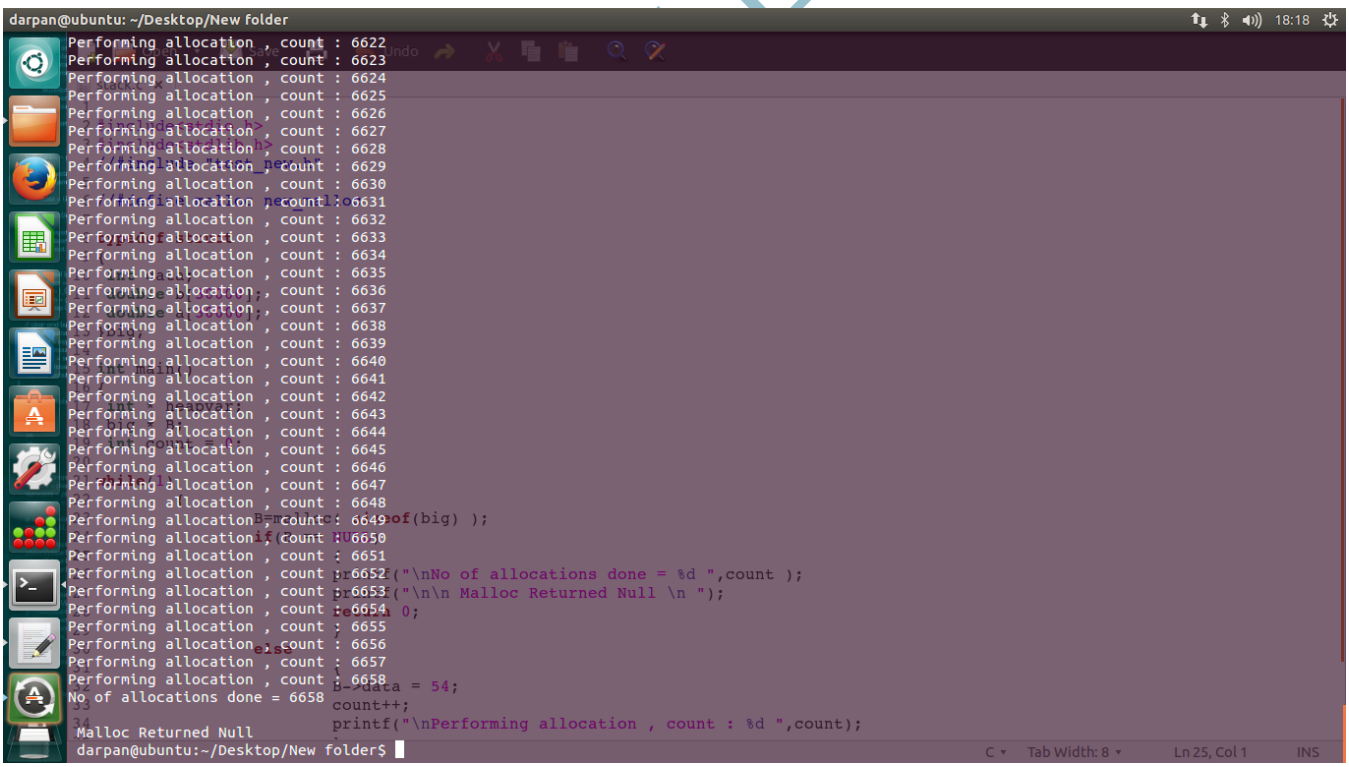
   c. Having got the information about what to search? And where to search?,
6. Our algorithm gets in to the mark phase first for finding and marking the garbage and then in to the sweep phase for collecting the garbage.
7. Mark Phase
   a. If an object is found to be alive then mark it. This is done by setting a field in the record-keeping node corresponding to this object.
8. Sweep Phase
   a. Scan all the objects. If an object is marked then unmark it. If an object is not marked then free it as it has been declared as garbage by the mark phase of our collector.
9. Our algorithm
   a. Exits printing the message “Your request cannot be satisfied even after garbage collection”, if allocation fails even after trying to garbage collect.
   b. Returns the pointer to the allocated object of the size requested by the user if allocation was successful.

5. HOW TO USE OUR LIBRARY:-

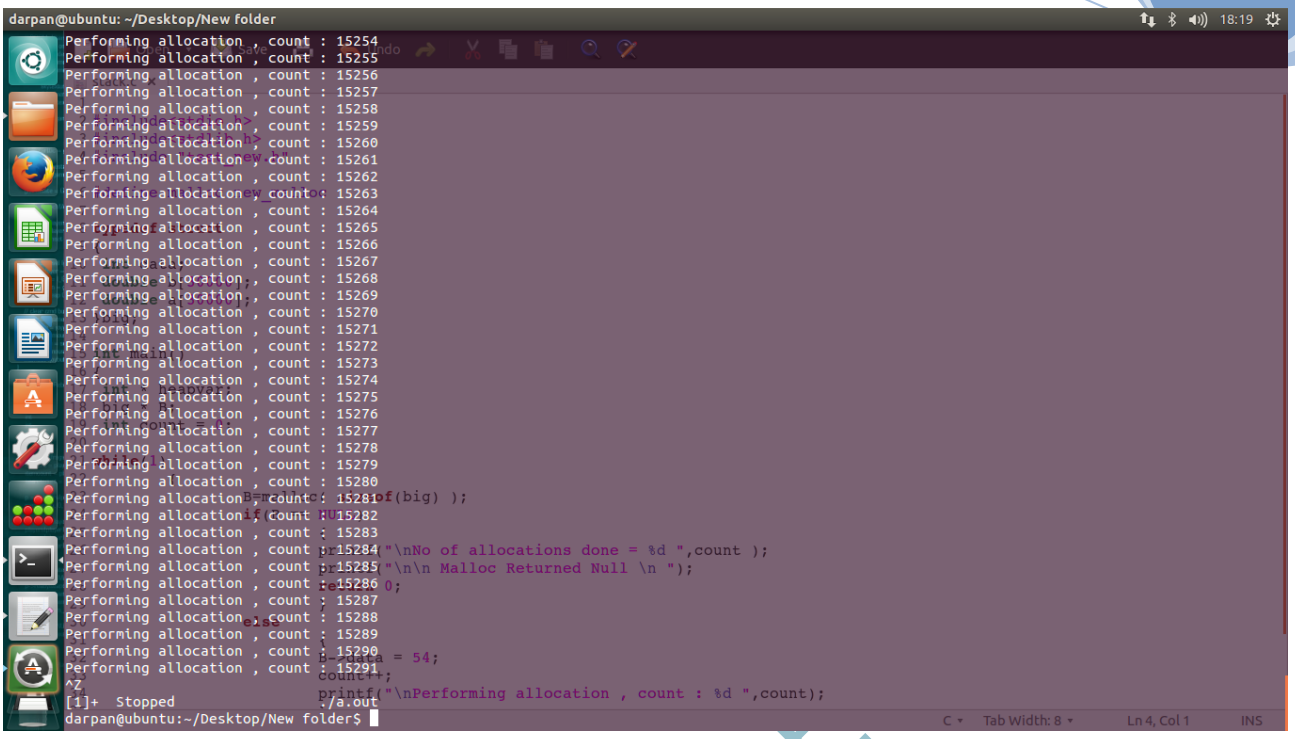
- Put our library "test_new.h" in the same folder where you are writing your c code.
- Include "test_new.h" in your c code like we include other libraries in the beginning.
- define malloc is equal to new_malloc
- And now you do not need to write free explicitely. Our collector will take care of memory



6. SCREENSHOTS OF THE OUTPUT:-



In this screen shot we can see that memory is allocated for only 6659 nodes without including our garbage collector. When program runs out of memory Null is returned.



In this screen shot we can see that after including our library, garbage is collected and memory is allocated to as many nodes as we want.

7. CONLUSION & LEARNING:-

- Fragmentation is a phenomenon that occurs in a long-running program that has undergone garbage collection several times. The problem is that objects tend to become spread out in the heap. Live objects end up being separated by many, small unused memory regions.
- The mark-and-sweep algorithm does not address fragmentation. Even after reclaiming the storage from all garbage objects, the heap may still be too fragmented to allocate the required amount of space.
- There is no final verdict on the suitability of GC for C programs.
- C is not compatible for kernel level implementation but research are going on to implement GC in C.
- Our garbage collection routine currently pauses the program when it runs.
- C language provides very little help in writing correct, bug free code. It doesn't have any standard for memory management.

8. REFERENCES:-

- http://programmers.stackexchange.com/questions/113177/why-do-languages-such-as-c-and-c-not-have-garbage-collection-while-java-does
- http://web.engr.illinois.edu/~maplant2/gc.html
- file:///C:/Users/sunil/Downloads/rick.pdf
- http://www.cs.utah.edu/~regehr/papers/ismm15-rafkind.pdf
- http://www.codeproject.com/Articles/31747/Conservative-Garbage-Collector-for-C
- http://www.linuxjournal.com/article/6679?page=0,1
- http://www.embedded.com/design/prototyping-and-development/4008335/Garbage-collection-in-C-language-code-applications
- http://www.utdallas.edu/~rsv031000/Projects/GC_for_C/User_Manual.pdf