</>

Memory Allocators

Section 1

Introduction

Applicable for

LINUX AND C

The Need for Memory Allocators

The Need for Memory Allocators

Initially memory allocation was done using system calls like ***brk()*** and ***sbrk().***

The Need for Memory Allocators

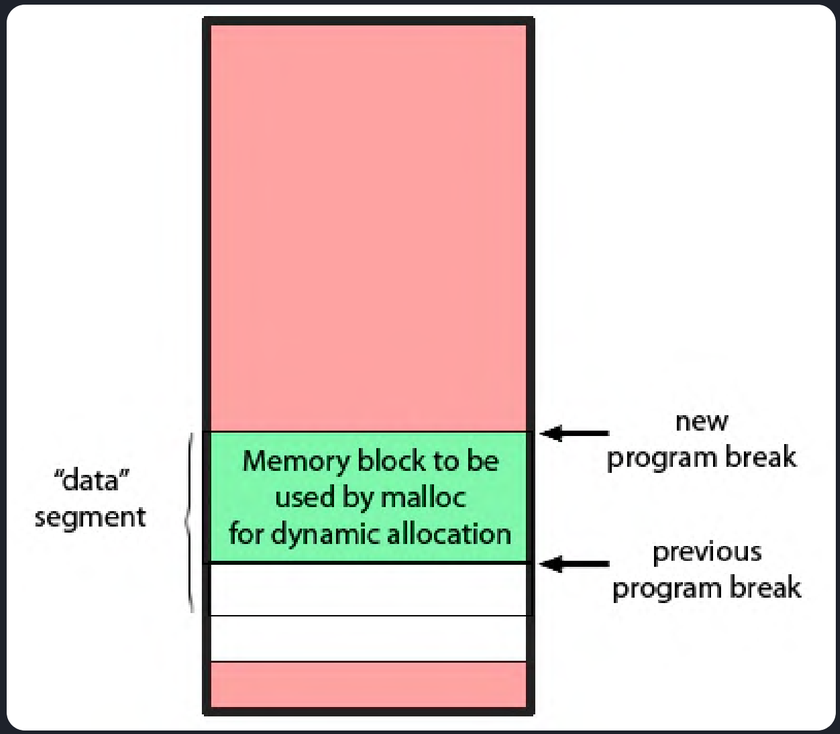
Initially memory allocation was done using system calls like ***brk()*** and ***sbrk().***

These calls increased/decreased the heap pointer to allocate memory from the program heap.

The Need for Memory Allocators

Initially memory allocation was done using system calls like ***brk()*** and ***sbrk().***

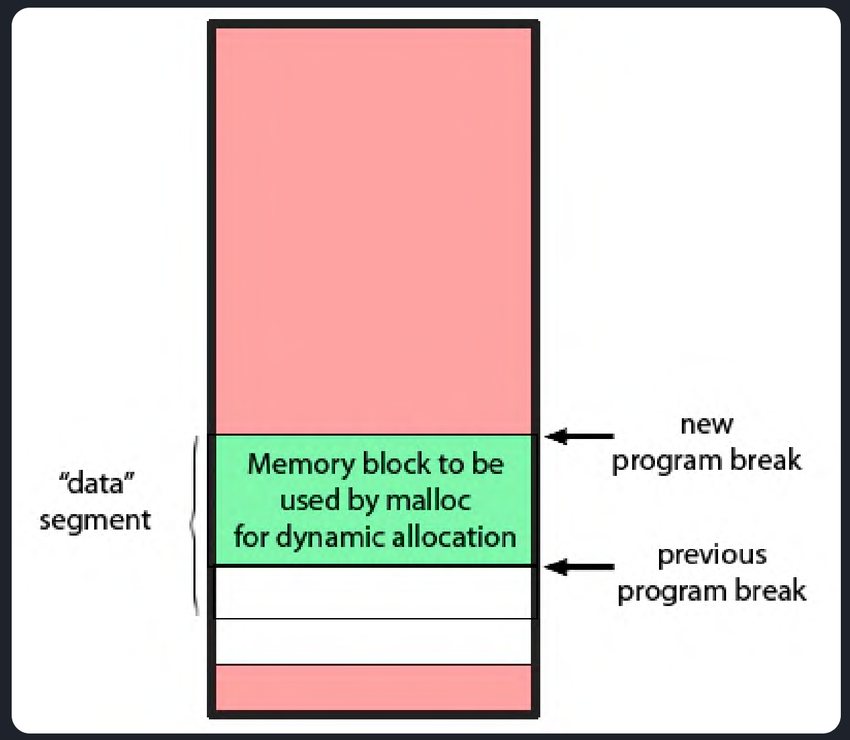
These calls increases/decreases the heap pointer to allocate memory from the program heap.



The Need for Memory Allocators

Initially memory allocation was done using system calls like ***brk()*** and ***sbrk().***

These calls increases/decreases the heap pointer to allocate memory from the program heap.




```c
void *malloc(size_t size)
{
        void *block;
        block = sbrk(size);
        if (block == (void*) -1)
                return NULL;
        return block;
}
```

The Need for Memory Allocators

They had few major disadvantages :

The Need for Memory Allocators

They had few major disadvantages :

- **syscalls overhead** - syscalls requires switching from user mode to kernel mode which has its own overhead. Allocating small sized memory using these functions can be inefficient.

The Need for Memory Allocators

They had few major disadvantages :

- **syscalls overhead** - syscalls requires switching from user mode to kernel mode which has its own overhead. Allocating small sized memory using these functions can be inefficient.
- **fragmentation** - The memory allocated may be fragmented. These heap based memory allocation do not give any way to use the fragmented memory and is thus wasted.

The Need for Memory Allocators

They had few major disadvantages :

- **syscalls overhead** - syscalls requires switching from user mode to kernel mode which has its own overhead. Allocating small sized memory using these functions can be inefficient.
- **fragmentation** - The memory allocated may be fragmented. These heap based memory allocation do not give any way to use the fragmented memory and is thus wasted.
- **Copy on write (COW)** - Unix based operating systems generally employ copy-on-write mechanism for virtual memory management, specially in cases like fork(). So it may be possible that the memory requested has been marked as *not present* and accessing that memory can cause page fault issues.

Possible quick explanation example and explanation CoW if people want it at the end.

Dynamic Storage Allocation: A Survey and Critical Review * **

Paul R. Wilson, Mark S. Johnstone, Michael Neely, and David Boles***

Department of Computer Sciences
University of Texas at Austin
Austin, Texas, 78751, USA
(wilson|markj|neely@cs.utexas.edu)

Abstract. Dynamic memory allocation has been a fundamental part of most com-

1 Introduction

Paul Wilson and his colleagues have written an excellent survey paper on allocation techniques that discusses some of the goals in more detail - "Dynamic Storage Allocation: A Survey and Critical Review" in *International Workshop on Memory Management*, September 1995.

Conclusion : They discuss, minimizing space by minimizing wastage (generally due to fragmentation) must be the primary goal in any allocator.

</>

Implementations

</>

dlmalloc

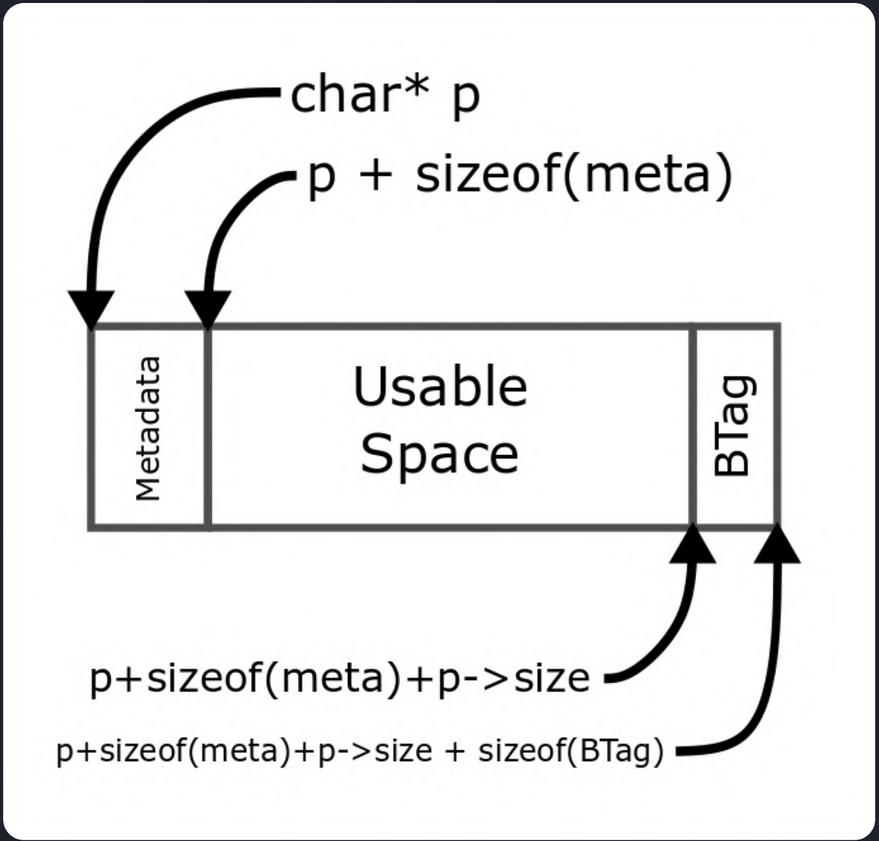
A MEMORY ALLOCATOR BY ***DOUG LEA***

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.

dlmalloc - Algorithms

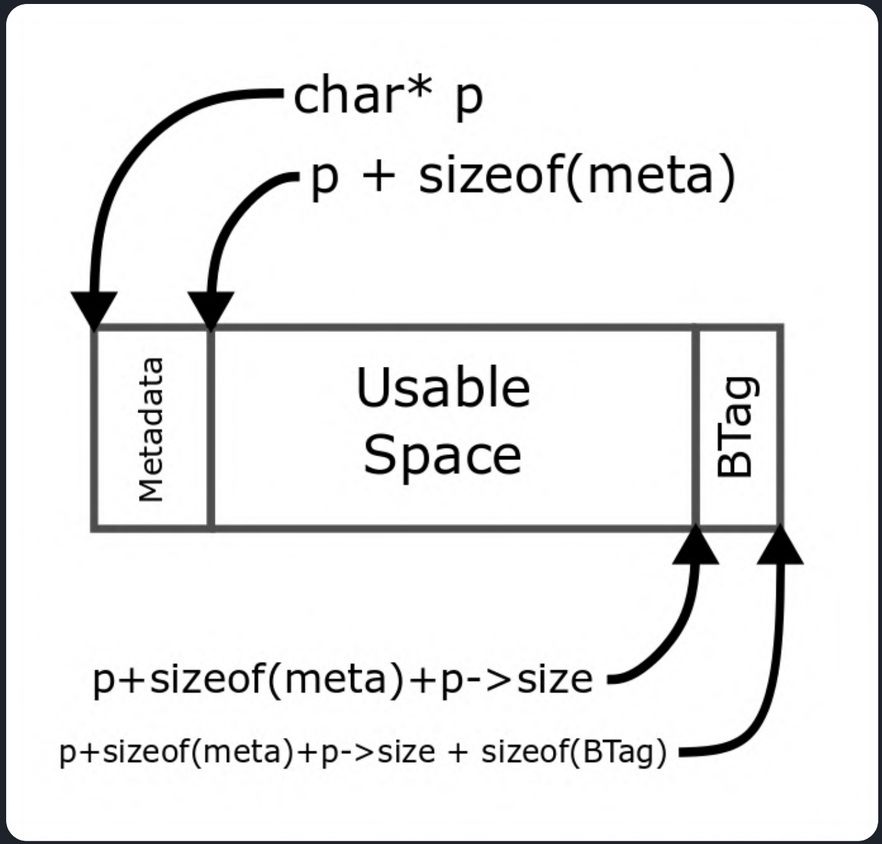
Even though the program has been revised and improved over the years, but 2 core algorithms are still present.



Boundary Tags :

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.

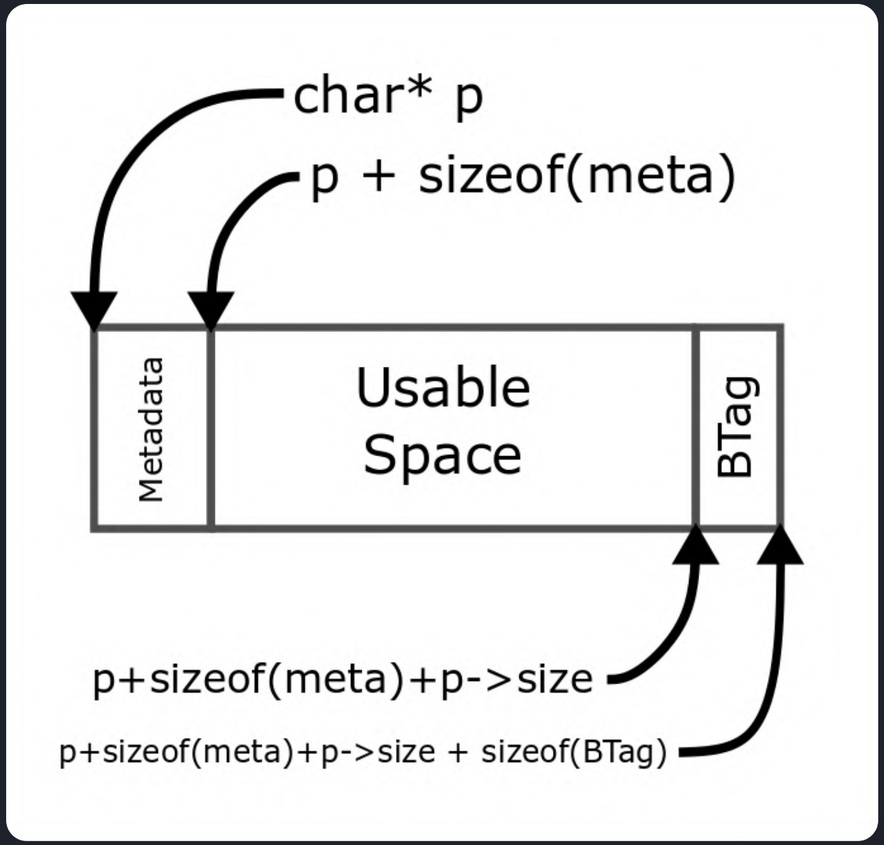


Boundary Tags :

Chunks of memory carry around with them size information fields both before and after the chunk.

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.



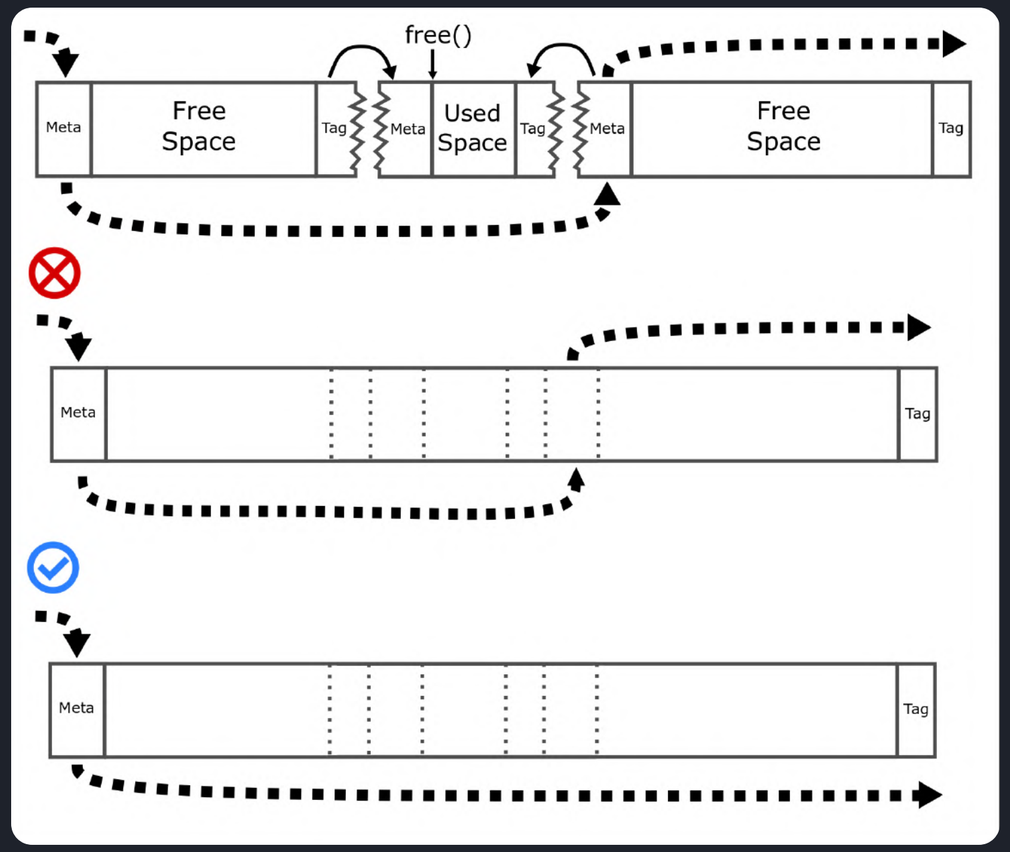
Boundary Tags :

Chunks of memory carry around with them size information fields both before and after the chunk.

This allows for two important capabilities:

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.



Boundary Tags :

Chunks of memory carry around with them size information fields both before and after the chunk.

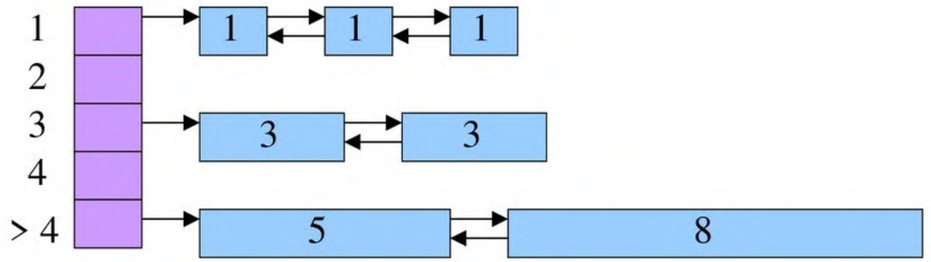
This allows for two important capabilities:

- Two bordering unused chunks can be coalesced into one larger chunk. This minimizes the number of unusable small chunks.
- All chunks can be traversed starting from any known chunk in either a forward or backward direction.

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.

- Have a bin for each block size, up to a limit
  - Advantages: no search for requests up to that size
  - Disadvantages: many bins, each storing a pointer
- Except for a final bin for all larger free blocks
  - For allocating larger amounts of memory
  - For splitting to create smaller blocks, when needed

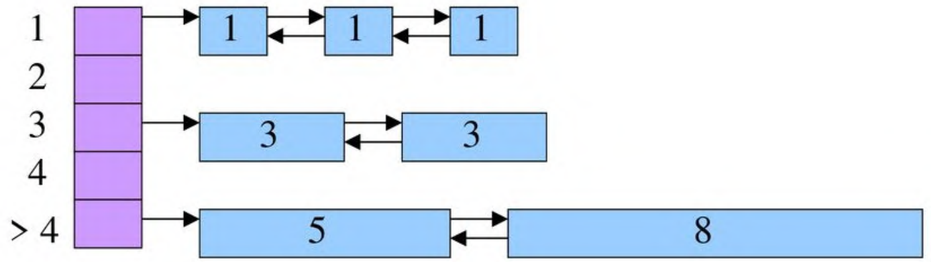


Binning :

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.

- Have a bin for each block size, up to a limit
  - Advantages: no search for requests up to that size
  - Disadvantages: many bins, each storing a pointer
- Except for a final bin for all larger free blocks
  - For allocating larger amounts of memory
  - For splitting to create smaller blocks, when needed



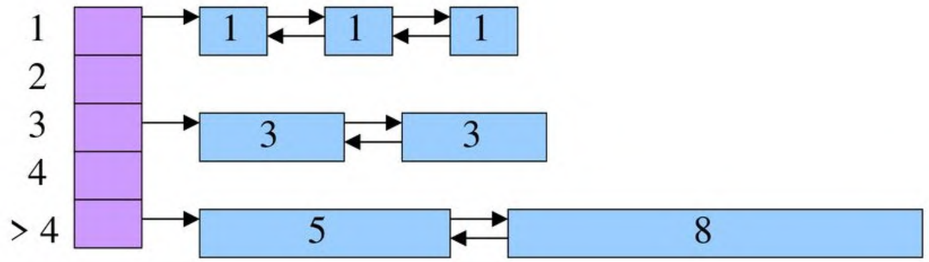
Binning :

A linked list is used to store free spaces in order of power of 2 to avoid unnecessary breaking of large blocks everytime.

dlmalloc - Algorithms

Even though the program has been revised and improved over the years, but 2 core algorithms are still present.

- Have a bin for each block size, up to a limit
  - Advantages: no search for requests up to that size
  - Disadvantages: many bins, each storing a pointer
- Except for a final bin for all larger free blocks
  - For allocating larger amounts of memory
  - For splitting to create smaller blocks, when needed



Binning :

A linked list is used to store free spaces in order of power of 2 to avoid unnecessary breaking of large blocks everytime.

Best fit strategy (searching if the required size block exists in the bin else breaking a large block into that size) is used to allocate memory for new allocation requests.

dlmalloc

These two algorithms are used to allocate and free memory blocks in `malloc()`.

SOURCE BLOG: http://gee.cs.oswego.edu/dl/html/malloc.html

SOURCE CODE: http://gee.cs.oswego.edu/pub/misc/malloc.c

dlmalloc

These two algorithms are used to allocate and free memory blocks in `malloc()`.

Malloc uses `sbrk()` for allocation of small blocks.

SOURCE BLOG: http://gee.cs.oswego.edu/dl/html/malloc.html

SOURCE CODE: http://gee.cs.oswego.edu/pub/misc/malloc.c

dlmalloc

These two algorithms are used to allocate and free memory blocks in `malloc()`.
Malloc uses `sbrk()` for allocation of small blocks.
For larger blocks it uses a relatively new method : `mmap()`. It provides the benefit that it maps memories without **holes**. The disadvantage it has that it is sloooower than sbrk(I have yet to study mmap)

SOURCE BLOG: http://gee.cs.oswego.edu/dl/html/malloc.html

SOURCE CODE: http://gee.cs.oswego.edu/pub/misc/malloc.c

dlmalloc

These two algorithms are used to allocate and free memory blocks in `malloc()`.
Malloc uses `sbrk()` for allocation of small blocks.
For larger blocks it uses a relatively new method : `mmap()`. It provides the benefit that it maps memories without **holes**. The disadvantage it has that it is sloooower than sbrk(I have yet to study mmap).

As more contributions were made to the code. It has improved much more and introduced new feature which you can refer to like : *Locality preservation, Wilderness Preservation, Caching.*

SOURCE BLOG: http://gee.cs.oswego.edu/dl/html/malloc.html
SOURCE CODE: http://gee.cs.oswego.edu/pub/misc/malloc.c

dlmalloc

These two algorithms are used to allocate and free memory blocks in `malloc()`.
Malloc uses `sbrk()` for allocation of small blocks.
For larger blocks it uses a relatively new method : `mmap()`. It provides the benefit that it maps memories without **holes**. The disadvantage it has that it is slooooower than sbrk(I have yet to study mmap).

As more contributions were made to the code. It has improved much more and introduced new feature which you can refer to like : *Locality preservation, Wilderness Preservation, Caching.*

A big disadvantage : dlmalloc works only for single threaded programs. It fails for multi-threaded programs.

SOURCE BLOG: http://gee.cs.oswego.edu/dl/html/malloc.html
SOURCE CODE: http://gee.cs.oswego.edu/pub/misc/malloc.c

</>

ptmalloc

THE GLIBC IMPLEMENTATION

ptmalloc

ptmalloc

ptmalloc - **pthreads malloc** is the current glibc implementation for malloc. It is a derived version of ***dlmalloc*** except that it has a **pthreads** wrapper which helps malloc support multithreaded programs for memory allocation.

ptmalloc

ptmalloc - **pthreads malloc** is the current glibc implementation for malloc. It is a derived version of ***dlmalloc*** except that it has a **pthreads** wrapper which helps malloc support multithreaded programs for memory allocation.

IMPLEMENTATION DETAILS : https://sourceware.org/glibc/wiki/MallocInternals

SOURCE CODE : https://github.com/hustfisher/ptmalloc

THANK YOU - PRESENTATION BY

Mohammad Shehar Yaar Tausif

https://github.com/sheharyaar

+91-9934180769

+91-9934180769

sheharyaar48@gmail.com

EARNING? YOU CAN SUPPORT ME TOO :)

https://paypal.me/lagn0s

9934180769@paytm

+91-9934180769