Next, previous, before and after

Daniel Gustafsson

2020-01-13

1 Introduction

Everyone who has ever tried programming in C should have heard of "malloc" and "free", these two functions provide a way for the programmer to easily allocate memory on the heap of a process. Modern day memory management is fast and efficient, but what acctually goes on under the hood of the operating system when a process calls "malloc"? In this report I will create my own heap manager similar to "malloc" and explore different aspects approaches, can i make it really space-efficient? Will it have a speed payoff? Continue reading and you will find answers.

2 Background

To create your own heap manager, you first need to understand how a heap manager works. The traditional way is to have a linked list data-structure where you keep all the free blocks from your heap, then when the process wants to allocate something the manager finds a free block with appropriate size and returns the virtual memory address. Later when (hopefully) "free" is called by the process, the manager will put the newly freed block back in the list of free blocks. To make this possible we need to implement a couple things. Firstly we need to get memory from the operating system , we can do this with the "mmap" system call, we should get as much memory as we think the process will use right from the start because calling the system to get more all the time would be very performance costly. Now we need a header with some information to be stored in every block on the heap. To start, the header should contain the following information,

- Free - if the block is free or taken
- Size - the size of the block
- Bfree - if the block before is free or taken
- Bsize - the size of the block before
- Next - pointer to the next block in the list
- Prev - pointer to the previous block in the list

When a block is requested by the process, we need to make sure the process wont get a block that is

3 Method

this is how i did it

4 Results

this is the results