Jonathan Parlett

May 30, 2022

Memory in C

What were going to talk about

- Segmentation of memory. How computer memory is allocated by the operating system

What were going to talk about

- Segmentation of memory. How computer memory is allocated by the operating system
- What is a segmentation fault

What were going to talk about

- Segmentation of memory. How computer memory is allocated by the operating system
- What is a segmentation fault
- The Stack and the Heap

What were going to talk about

- Segmentation of memory. How computer memory is allocated by the operating system
- What is a segmentation fault
- The Stack and the Heap
- Malloc, how we request memory from the operating system

What were going to talk about

- Segmentation of memory. How computer memory is allocated by the operating system
- What is a segmentation fault
- The Stack and the Heap
- Malloc, how we request memory from the operating system
- Static vs Dynamic Memory, and why you would use one over the other

Segmentation of Memory

A big part of C is memory management, so lets discusses how memory is relevant to programs.

- Each program has a section of memory allocated to it by the operating system when it is executed.

Segmentation of Memory

A big part of C is memory management, so lets discusses how memory is relevant to programs.

- Each program has a section of memory allocated to it by the operating system when it is executed.
- Data items and structures such as, arrays, variables and pointers declared in your program are stored in this section of memory.

Segmentation of Memory

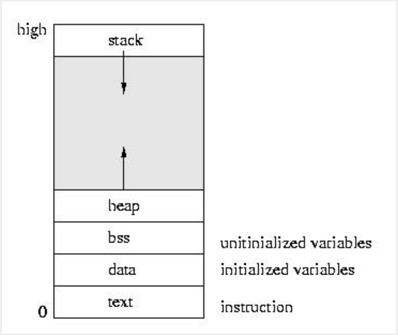
A big part of C is memory management, so lets discusses how memory is relevant to programs.

- Each program has a section of memory allocated to it by the operating system when it is executed.
- Data items and structures such as, arrays, variables and pointers declared in your program are stored in this section of memory.
- Programs are not allowed to touch memory outside of what has been allocated by the operating system. If they do a segmentation fault occurs.

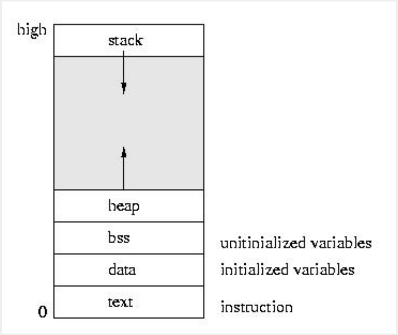
Stack and Heap

- The memory allocated to a C program is further divided into the **Stack** and the **Heap**.

Stack and Heap

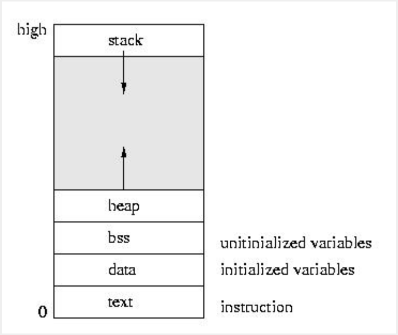
- The memory allocated to a C program is further divided into the **Stack** and the **Heap**.
- 

Stack and Heap

- The memory allocated to a C program is further divided into the **Stack** and the **Heap**.
- 

- As data is stored on the Stack it grows down in memory towards the Heap. Increasingly higher valued addresses are used.

Stack and Heap

- The memory allocated to a C program is further divided into the **Stack** and the **Heap**.
- 

- As data is stored on the Stack it grows down in memory towards the Heap. Increasingly higher valued addresses are used.
- As data is stored on the Heap it grows up in memory towards the Stack. Increasingly lower valued addresses are used.

Malloc

- This is a brief overview of malloc, if you don’t remember its use from CS265 don’t be concerned, we’ll cover its use in more detail later.

Malloc

- This is a brief overview of malloc, if you don’t remember its use from CS265 don’t be concerned, we’ll cover its use in more detail later.
- Malloc is used to request memory from the operating system. Specifically it requests memory from the Heap, not the stack.

Malloc

- This is a brief overview of malloc, if you don’t remember its use from CS265 don’t be concerned, we’ll cover its use in more detail later.
- Malloc is used to request memory from the operating system. Specifically it requests memory from the Heap, not the stack.
- `malloc(5)`

  will return 5 bytes on the heap for example.

Malloc

- This is a brief overview of malloc, if you don't remember its use from CS265 don't be concerned, we'll cover its use in more detail later.
- Malloc is used to request memory from the operating system. Specifically it requests memory from the Heap, not the stack.
-
  ```
  malloc(5)
  ```
  will return 5 bytes on the heap for example.
- We can use the memory given by malloc to store data on the heap.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
    - Requesting it from the operation system I.E *malloc*.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
  - Requesting it from the operation system I.E *malloc*.
  - Freeing it so it can be reused later.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
  - Requesting it from the operation system I.E *malloc*.
  - Freeing it so it can be reused later.
- These tasks are usually handled for you in higher level languages such as python.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
  - Requesting it from the operation system I.E *malloc*.
  - Freeing it so it can be reused later.
- These tasks are usually handled for you in higher level languages such as python.
- The most challenging, and useful thing about C is that you have the ability to fully control memory.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
  - Requesting it from the operation system I.E *malloc*.
  - Freeing it so it can be reused later.
- These tasks are usually handled for you in higher level languages such as python.
- The most challenging, and useful thing about C is that you have the ability to fully control memory.
- You also have the option of letting the compiler control it for you.

Static and Dynamic Memory

- All data is stored in memory. This is true for any programming language.
- There are a few maintenance tasks that come with using memory.
    - Requesting it from the operation system I.E *malloc*.
    - Freeing it so it can be reused later.
- These tasks are usually handled for you in higher level languages such as python.
- The most challenging, and useful thing about C is that you have the ability to fully control memory.
- You also have the option of letting the compiler control it for you.
- These two dynamics exposed in C are called Static and Dynamic Memory.

Static Memory

- Static memory, also called compile time memory is memory that is managed for you, by the compiler.

Static Memory

- Static memory, also called compile time memory is memory that is managed for you, by the compiler.
- It uses the Stack for storing data.

Static Memory

- Static memory, also called compile time memory is memory that is managed for you, by the compiler.
- It uses the Stack for storing data.
- When you are declaring variables, arrays, and structs, these data items are stored on the Stack.

Static Memory

- Static memory, also called compile time memory is memory that is managed for you, by the compiler.
- It uses the Stack for storing data.
- When you are declaring variables, arrays, and structs, these data items are stored on the Stack.
- The memory is allocated for them by the compiler when they come into scope, and it is deallocated when they fall out of scope.

Static Memory

- Static memory, also called compile time memory is memory that is managed for you, by the compiler.
- It uses the Stack for storing data.
- When you are declaring variables, arrays, and structs, these data items are stored on the Stack.
- The memory is allocated for them by the compiler when they come into scope, and it is deallocated when they fall out of scope.
- These variables would be statically allocated on the stack

```c
int i = 0;
int arr[10];
```

Dynamic Memory

- Dynamic memory is the memory you control. You are responsible for requesting it from the operating system, and saying when it is no longer needed so it can be reused.

Dynamic Memory

- Dynamic memory is the memory you control. You are responsible for requesting it from the operating system, and saying when it is no longer needed so it can be reused.
- Dynamic memory uses the Heap.

Dynamic Memory

- Dynamic memory is the memory you control. You are responsible for requesting it from the operating system, and saying when it is no longer needed so it can be reused.
- Dynamic memory uses the Heap.
- Dynamic memory is powerful because it allows you to utilize memory programmatically.

Dynamic Memory

- Dynamic memory is the memory you control. You are responsible for requesting it from the operating system, and saying when it is no longer needed so it can be reused.
- Dynamic memory uses the Heap.
- Dynamic memory is powerful because it allows you to utilize memory programmatically.
- You can create data structures that grow and shrink in response to different situations in your program.

Dynamic Memory

- Dynamic memory is the memory you control. You are responsible for requesting it from the operating system, and saying when it is no longer needed so it can be reused.
- Dynamic memory uses the Heap.
- Dynamic memory is powerful because it allows you to utilize memory programmatically.
- You can create data structures that grow and shrink in response to different situations in your program.
- It is what allows C to outperform other higher level languages. Greater control means greater optimization opportunities.

Static vs Dynamic Memory

- Static memory is simpler to use over dynamic memory. No need to worry about allocating or deallocating means less programmer overhead.
- Static memory is limited. We cannot adjust the amount of memory allocated for a static structure such as an array. This potentially creates a lot of wasted space.
- Dynamic memory is complicated. There is a lot of opportunities to make mistakes.
- Dynamic memory is flexible. We can adjust the amount of memory allocated to a structure on the fly.
- Dynamic memory also allows us to be far more efficient then higher level languages.

Thanks For Watching

Hopefully you've gained a high level overview of the memory structure of a C program and are prepared to go more in depth into memory and pointers.