Dynamic memory allocation Memory management, Types of memory allocation, Allocation (malloc, calloc, realloc), Deallocation(free)

Dynamic memory allocation

Dynamic memory allocation is the process of assigning the memory space during the execution time or the run time. It is allocation and deallocation of memory at run-time.

Characteristics

1. When we do not know how much amount of memory would be needed for the program beforehand.
2. When we want data structures without any upper limit of memory space.
3. When you want to use your memory space more efficiently.Example: If you have allocated memory space for a 1D array as array[20] and you end up using only 10 memory spaces then the remaining 10 memory spaces would be wasted and this wasted memory cannot even be utilized by other program variables.
4. Dynamically created lists insertions and deletions can be done very easily just by the manipulation of addresses whereas in case of statically allocated memory insertions and deletions lead to more movements and wastage of memory.
5. When you want you to use the concept of structures and linked list in programming, dynamic memory allocation is a must.

Memory management

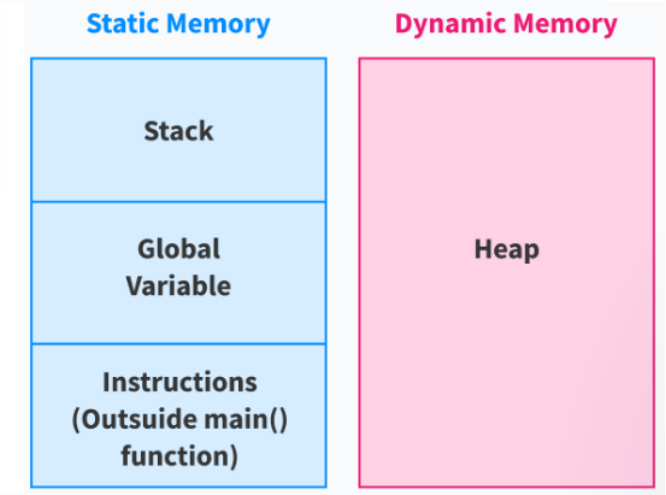
two types of memory in our machine, one is Static Memory and another one is Dynamic Memory; both the memory are managed by the Operating System. The operating system helps in the allocation and deallocation of memory blocks either during compile-time or during the run-time of our program.
When the memory is allocated during compile-time it is stored in the Static Memory and it is known as Static Memory Allocation, and when the memory is allocated during run-time it is stored in the Dynamic Memory and it is known as Dynamic Memory Allocation.

Static Memory Allocation	Dynamic Memory Allocation
Constant (Invariable) memory is reserved at **compile-time** of our program that **can't** be modified.	Dynamic (Variable) memory is reserved at **run-time** of our program that **can** be modified.
It is used at compile-time of our program and is also known as **compile-time memory allocation**.	It is used at run-time of our program and is also known as **run-time memory allocation**.
We **can't** allocate or deallocate a memory block during run-time.	We **can** allocate and deallocate a memory block during run-time.
Stack space is used in Static Memory Allocation.	Heap space is used in Dynamic Memory Allocation.

It **doesn't provide** reusability of memory, while the program is running. So, it is **less** efficient.	It **provides** reusability of memory, while the program is running. So, it is **more** efficient.

During execution some memory is required to store its variables, functions, instructions and the program file itself. The system memory is divided into four

1. Stack Segment (Static Memory)
2. Global Variables Segment (Static Memory)
3. Instructions / Text Segment (Static Memory)
4. Heap Segment (Dynamic Memory)



The amount of memory allocated for Stack,
Global Variables, and Instructions / Text during compile-time is invariable and cannot be reused until the program execution finishes. However, the Heap segment of the memory can be used at run-time and can be expanded until the system's memory exhausts.

Instructions / Text

- Text outside the main() function is stored in the **Static Memory**.
- These instructions are stored during compile-time of a program.

Global Variables

- Global variable also known as static variables and can be declared by two methods,
  - Using static keyword, ex. static int i = 0;
  - Declaring variable outside main() or any other function.
- These variables are stored in the Static Memory during the compile-time of our program.

Stack

- Stack is a constant space (memory) allocated by our operating system to store local variables, function calls and local statements that are present in the function definition. It is the major part of the **Static Memory** in our system.
- There are some drawbacks of stack space as follows:
  - The memory allocated for stack can't grow during the run-time of an application.
  - We can't allocate or deallocate memory during execution.

Heap

- Heap is the **Dynamic Memory** of our program, It can be also be imagined as a big free pool of memory available to us. The space occupied by heap section is not fixed, it can vary during the run-time of our program and there are functions to perform allocation and deallocation of memory blocks during run-time.

- Heap space can grow as long as we do not run out of the system's memory itself, however it is not in the best interest of a programmer to exhaust the system's memory, so we need to be really careful while using the heap space in our program.

Types of memory allocation

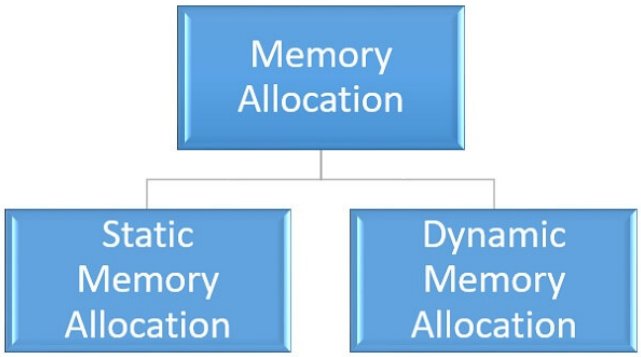
Memory Allocation

To get a process executed it must be first placed in the memory. Assigning space to a process in memory is called memory allocation. Memory allocation is a general aspect of the term **binding.**

Example a program has variable/attribute or set of attributes assigned with values. For storing these values, we must have memory allotted to these attributes.

So, the act of assigning the memory address to the attribute of the variable is called **memory allocation**. And the act of specifying/binding the values to the attributes of the variable is called **binding**. This action of binding must be performed before the variable is used during the execution of the program.

We have two types of memory allocation or we can say two methods of binding, static and dynamic binding.



Types of Memory Allocation

1. Static Memory Allocation

Static memory allocation is performed when the compiler compiles the program and generates object files. The linker merges all these object files and creates a single executable file. The loader loads this single executable file in the main memory, for execution. In static memory allocation, the size of the data required by the process must be known **before** the execution of the process initiates.

If the data sizes are not known before the execution of the process, then they have to be guessed. If the data size guessed is larger than the required, then it leads to **wastage** of memory. If the guessed size is smaller, then it leads to inappropriate execution of the process.

The static memory allocation method does not need any memory allocation operation during the execution of the process. All the memory allocation operation required for the process is done before the execution of the process has started. So, it leads to **faster** execution of a process.

Static memory allocation provides more **efficiency** when compared to dynamic memory allocation.

2. Dynamic Memory Allocation

Dynamic memory allocation is performed while the program is in execution. Here, the memory is allocated to the entities of the program when they are to be used for the **first time** while the program is running.

The actual size, of the data required, is known at the run time so, it allocates the **exact** memory space to the program thereby, reducing the memory wastage.

Dynamic memory allocation provides **flexibility** to the execution of the program. As it can decide what amount of memory space will be required by the program. If the program is large enough then a dynamic memory allocation is performed on the different parts of the program, which is to be used currently. This reduces memory wastage and improves the performance of the system.

Allocating memory dynamically creates an overhead over the system. Some allocation operations are performed repeatedly during the program execution creating more overheads, leading in **slow** execution of the program.

Thus the dynamic memory allocation is flexible but slower than static memory allocation.

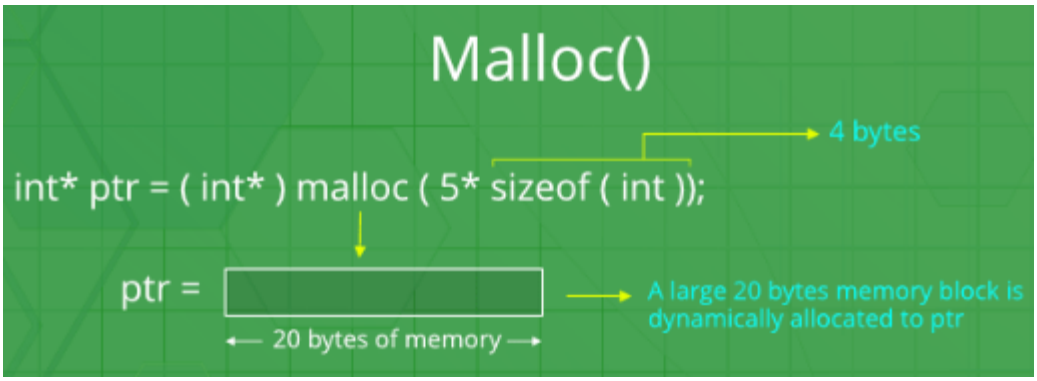
Allocation (malloc, calloc, realloc), Deallocation(free)

Malloc

The **"malloc"** or **"memory allocation"** method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It doesn't Initialize memory at execution time so that it has initialized each block with the default garbage value initially.

Syntax:
ptr = (cast-type*) malloc(byte-size)

For Example:
ptr = (int*) malloc(100 * sizeof(int));
Since the size of int is 4 bytes, this statement will allocate 400 bytes of memory. And, the pointer ptr holds the address of the first byte in the allocated memory.



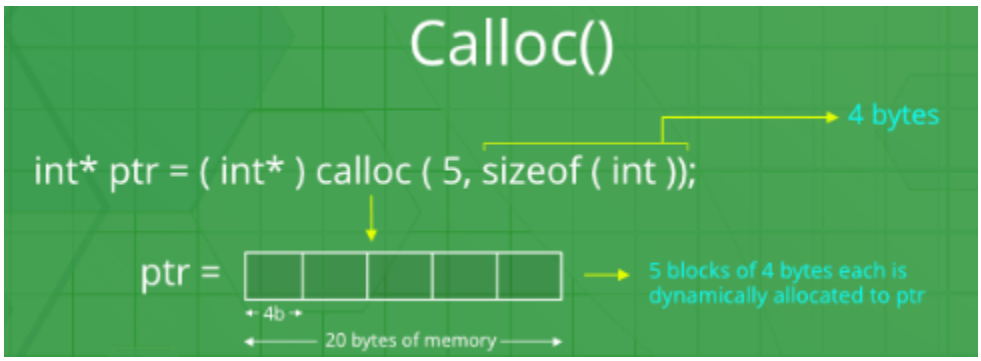
Calloc

1. **"calloc"** or **"contiguous allocation"** method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. it is very much similar to malloc() but has two different points and these are:
2. It initializes each block with a default value '0'.
3. It has two parameters or arguments as compare to malloc().
   Syntax:
   ptr = (cast-type*)calloc(n, element-size);

   here, n is the no. of elements and element-size is the size of each element.

   For Example:

   ptr = (float*) calloc(25, sizeof(float));
   This statement allocates contiguous space in memory for 25 elements each with the size of the float.

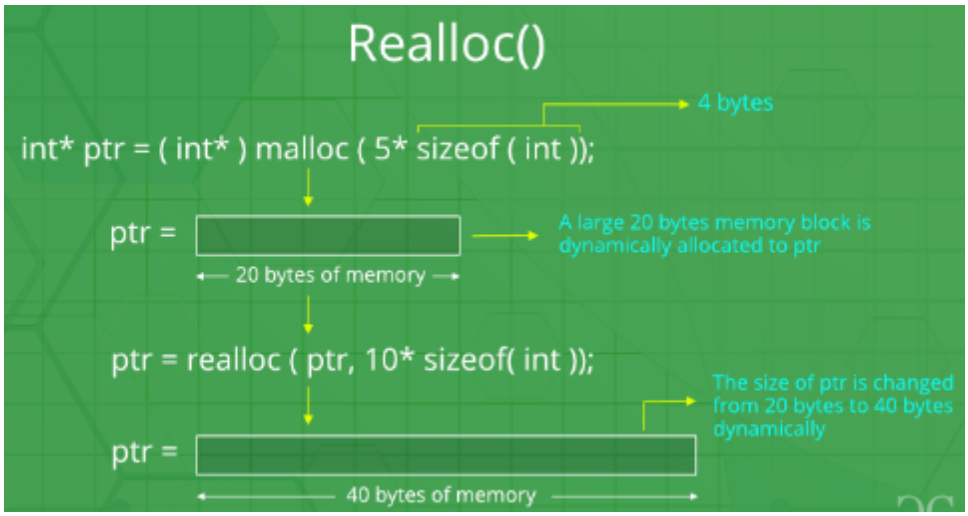


Realloc

“realloc” or **“re-allocation”** method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to **dynamically re-allocate memory**. re-allocation of memory maintains the already present value and new blocks will be initialized with the default garbage value.

Syntax:
ptr = realloc(ptr, newSize);

where ptr is reallocated with new size 'newSize'.



Deallocation(free)

“free” method in C is used to dynamically **de-allocate** the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on their own. Hence the free() method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

Syntax:
free(ptr);