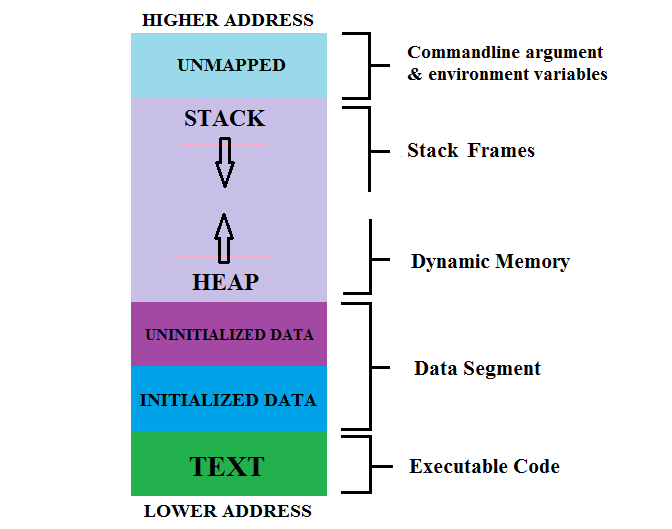
**Memory Layout of C Program**

**What is memory layout of C-program ?**

In practical words, when you run any C-program, its executable image is loaded into RAM of  computer in an organized manner which is called process address space or Memory layout of

CProgram.  
This memory layout is organized in following fashion:

* Text segment
* Data segment
* Heap segment
* Stack segment
* Unmapped or reserved



**Text segment**

Text segment contain executable instructions of your C program, its also called code segment. This is the machine language representation of the program steps to be carried out, including all functions making up the program, both user defined and system. The text segment is sharable so that only a single copy needs to be in memory for different executing programs, such as text editors, shells, and so on. Usually, text segment is read-only, to prevent a program from accidentally modifying its instructions

**Data segment**

There are two sub section of this segment called initialized & uninitialized data segment

* **Initialized data:-** It contains both static and global data that are initialized with non-zero values.This segment can be further classified into *read-only* area and*read-write* area.

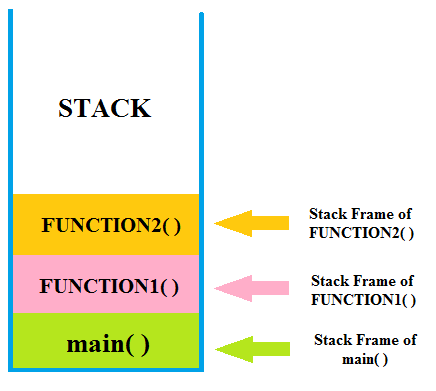
For example : The global string defined by char string[ ] = “hello world” and a statement like int count=1 outside the main (i.e. global) would be stored in initialized read-write area. And a global statement like const int A=3 makes the variable ‘A’ read-only and to be stored in initialized read-only area.

* **Uninitialized data (bss segment):-**Uninitialized data segment is also called BSS segment. BSS stands for ‘Block Started by Symbol’ named after an ancient assembler operator. Uninitialized data segment contains all global and static variables that are initialized to zero or do not have explicit initialization in source code.

For example : The global variable declared as int A would be stored in uninitialized data segment. A statement like static int X=0 will also stored in this segment cause it initialized with zero.

**Heap segment**

* The heap segment is area where dynamically allocated memory (allocated by malloc(), calloc(), realloc() and new for C++) resides.
* When we allocate memory through dynamic allocation techniques(in simple word, run time memory allocation), program acquire space from system and process address space grows that’s why we saw upward arrow indication in figure for Heap.
* We can free dynamically allocated memory space (by using Free() or delete). Freed memory goes back to the heap but doesn’t have to be returned to system (it doesn’t have to be returned at all), so unordered malloc/free’s eventually cause *heap fragmentation*.
* When we use dynamic allocation for acquire memory space we must keep track of allocated memory by using its address.
* Memory leak error is cause by excess use of dynamic allocation or Heap fragmentation



The stack segment is area where local variables are stored. By saying local variable means that all those variables which are declared in every function including main( ) in your C program.

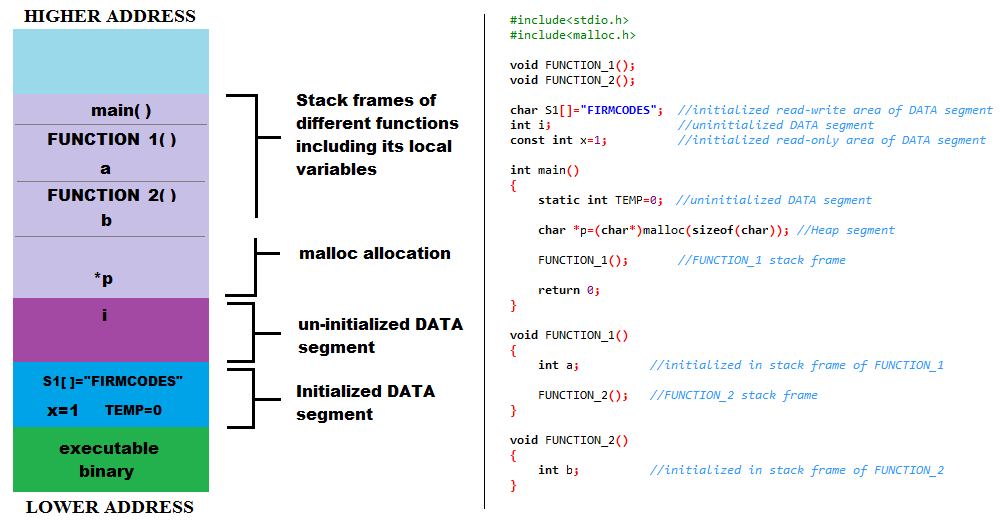
            When we call any function, stack frame is created and when function returns, stack frame is destroyed including all local variables of that particular function.

Stack frame contain some data like return address, arguments passed to it, local variables, and any other information needed by the invoked function.

              A “stack pointer (SP)” keeps track of stack by each push & pop operation onto it, by adjusted stack pointer to next or previous address.

**Unmapped *or* reservedsegment**

Unmapped or reserved segment contain command line arguments and other program related data like lower address-higher address of executable image, etc.



As you can see on above figure(click on it for large view), we write a simple code in which we use different variable declaration and three function including main( ) function on right half pic. And on left half of pic, i shown you that how this C program going to store in RAM. Left half of this pic is memroy layout of this C program.

As we discussed in previous tab of Memory layout of C program, your program is stored in memory(RAM) in particular fashion when it is executing and it divides in some blocks. Now we understand those blocks by using our example code seen above on right half of pic.

**Textsegment**

when we compile our code, we get executable code(which may be in any form like “.bin” , “.exe” and “.hex” , etc ) of our C program inform of output of compilaton. That particular executable code is goes to text segment and would be read only.

**Datasegment**

**Initialized Data segment:-**

* A string char S1[ ]=”FIRMCODES” and static TEMP=0 will store in initialized read-write area because we don’t use keyword like const or any other which makes variable read-only.
* Keyword static make any variable alive even if its scope is not exist or in simple word, static keyword keeps variable value through out program life.
* Const int x=1 will store in read only area because keyword const make it read only portion and prevent any modification to it.

**Uninitialized Data segment:-**

In our program, int i declared global  goes to this area of storage because it is not initialized or initialized to zero.

**Heap segment**

* When you compile your program, memory space allocated by other than dynamic allocation (that all locals & global variables) is fixed at compile time, but dynamic memory is not fixed. We can declared and access memory on run time by dynamic allocation techniques that’s why memory allocated by malloc( ),calloc( ) or new is called dynamic memory or run time allocation.
* In example program, we allocate 1 byte dynamic memory using malloc function and stored its address in pointer ‘p’ to keep track of that memory or to access it. We neither freed that memory nor use it. We just want to show you that how that memory actually stored in run time.
* If you freed that memory, the space(here 1 byte) acquired by that memory allocation is not goes back to system and create Heap fragmentation block.

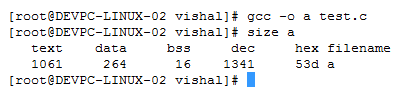
**Stack**

* Any C program contain atleast one main( ) function. So it creates stack frame cause it is also one kind of function. And rest of the all function is called(invoked in proper word) by main( ) function.
* Here we made three function named as main( ), FUNCTION\_1( ) and FUNCTION\_2( ) which is invoked nested, like main( ) invokes FUNCTION\_1( ) and then FUCNTION\_1( ) invokes FUNCTION\_2( ). All these function create its own stack frame along with its local variables as you can see on above pic.
* As the FUNCTION\_2( ) execution overs its local varible ‘b’ and its stack frame is destroyed, same as FUNCTION\_1( ) execution overs its local variable ‘a’ and its stack frame is destroyed, and same thing is done with your main( ) function execution.

**Unmappedorreservedsegment**

* As per our point of view, there is nothing in unmapped or reserved region because we dont pass any command line argument to our main.
* But system point of view, unmapped or reserved region contain program information like lower address, higher address and other program related data.

**Seethingspractically**



In this figure, we saw the code compilation of our example code(here name test.c), and then use size command to see the size of its executable file(here name ‘a’). And you can see the size of different blocks of text(executable code), data segment, bss, hex , dec segments.