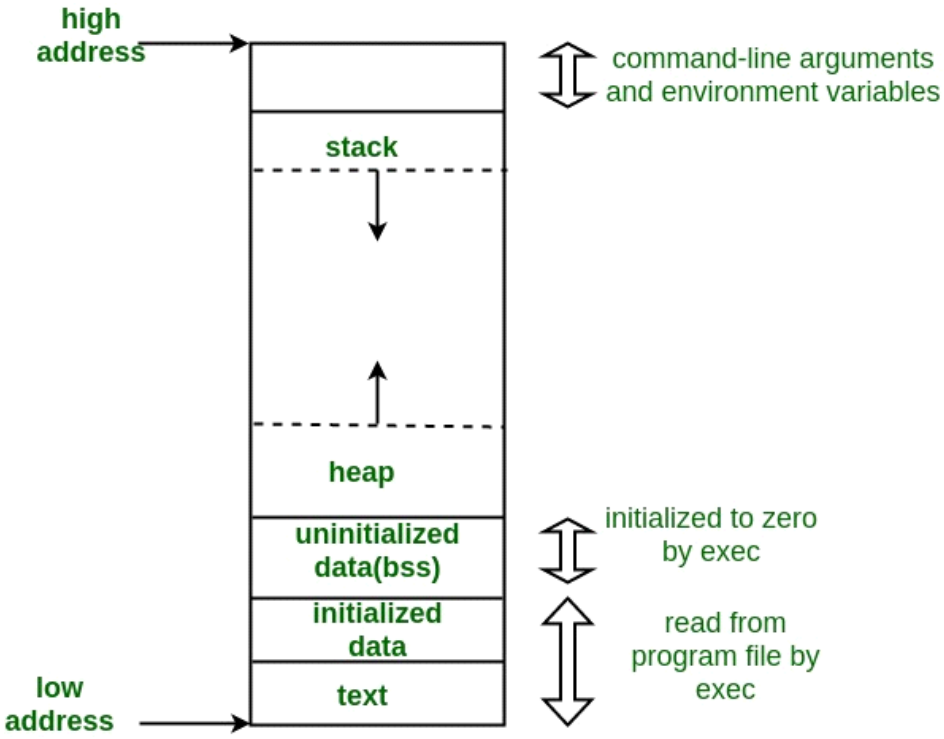
Memory Layout C

20 January 2022 20:24



Text Segment:

- Contains machine code of a compiled program, executable instructions
- Is sharable so that single copy needs to be in memory for frequently executed programs such as text editors, C Compilers
- Is read-only to prevent a program from accidentally overwrite the instructions
- Usually keep below heap or stack to prevent heap or stack overflows from overwriting it

Initialized Data Segment/Data Segment:

- It is a portion of virtual address space program
- Contains global, static, constant and extern variables
- Not read-only, as values of variables can be altered
- Further classified as:
    - Read-only area
    - Read-write area
- e.g. static int I = 1 || global int I = 10 - both are data segment

Uninitialized Data Segment(bss):

- Also known as "bss" segment, named after ancient assembler operator that stood for block started by symbol
- Starts at the end of data segment, contains all global, static variables which are initialized to 0 or don't have explicit initialization in source code
- Static int I or char c are examples

Heap:

- It is a segment where dynamic memory allocation happens, mostly using malloc, realloc, free and new
- Begins at the end of BSS segment i.e. uninitialized data segment and grow towards stack
- Shared by all shared libraries and dynamically loaded modules

Stack:

- Stack & Heap are traditionally located at the opposite ends of virtual address memory space program, when both meets free memory gets exhausted
- Contains LIFO structure, typically positioned at higher parts of memory
- A "stack pointer" register track of top of the stack, its value adjusted every time when a value is pushed on to the stack
- The set of values pushed for one function call is called "Stack Frame".
- Store all local variables and used to pass arguments to the function, along with the return address of the instruction to be executed after the function call is over.
- All recursive calls are added to stack.