# **Memory Layout:**

When the program runs, the processing is performed in two spaces called Kernel Spaceand User Spaceon the system. The two processing spaces implicitly interfere with each other and the processing of the program proceeds.

**Kernel Space**

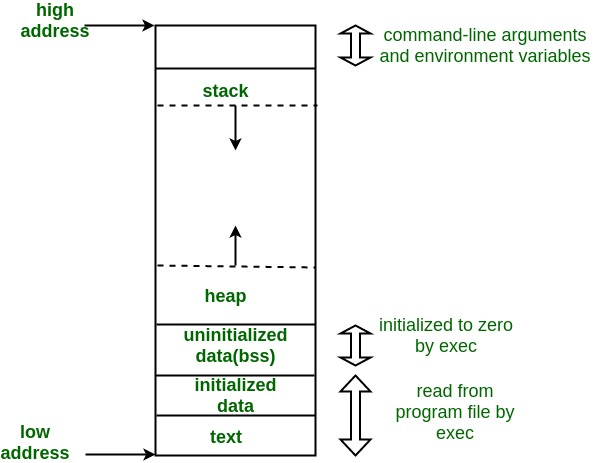
The kernel space can be accessed by user processes only through the use of system calls that are requests in a Unix-like operating system such as input/output (I/O) or process creation.

**User Space**

The user space is a computational resource allocated to a user, and it is a resource that the executing program can directly access. This space can be categorized into some segments.

When we executed any applications on the computer, the computer has to allocate memory for the program to run. The memory that is assigned to a program or application in a computer can be divided into five parts.

1. Text segment  
2. Initialized data segment  
3. Uninitialized data segment  
4. Stack  
5. Heap



1. Text Segment:

A text segment , also known as a code segment.

It contains executable instructions.

The text segment is often read-only, to prevent a program from accidentally modifying its instructions.

2. Initialized Data Segment:

It contains the global variables and static variables that are initialized by the programmer.

Ex:

#include <stdio.h>

**int** global = 10; /\* initialized global variable stored in DS\*/

**int** main(**void**)

{

**static** **int** i = 100; /\* Initialized static variable stored in DS\*/

**return** 0;

}

3. Uninitialized Data Segment:

The Uninitialized data segment called as “bss” (Block Started by Symbol**)** segment.

The data in this segment is initialized by the kernel to arithmetic 0 before the program starts executing.

Ex:

#include <stdio.h>

**int** global; /\* Uninitialized variable stored in bss\*/

**int** main(**void**)

{

**static** **int** i; /\* Uninitialized static variable stored in bss \*/

**return** 0;

}

4. Stack:

The stack is a segment of memory where data like your local variables and function calls get added and/or removed.

When you compile a program, the compiler enters through the main function and a stack frame is created on the stack.

The set of values pushed for one function call is termed a “stack frame”.

The automatic variables are stored, along with information that is saved each time a function is called.

The stack area traditionally adjoined the heap area and grew the opposite direction. When the stack pointer met the heap pointer, free memory was exhausted.

A “stack pointer” register tracks the top of the stack, it is adjusted each time a value is “pushed” onto the stack.

Mainly the following basic operations are performed in the stack:

* Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
* Pop: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.

## **Advantages of using Stack:**

* Stack automatically cleans up the object.
* A stack is used when a variable is not used outside that function.
* When a function is called the local variables are stored in a stack,

and it is automatically destroyed once returned.

* Not easily corrupted
* Variables cannot be resized.

**Stack overflow:**

Stack consist of a limited amount of [address space](https://en.wikipedia.org/wiki/Address_space), often determined at the start of the program. The size of the call stack depends on many factors, including the programming language, machine architecture, multi-threading, and amount of available memory. When a program attempts to use more space than is available on the call stack (that is, when it attempts to access memory beyond the call stack's bounds, which is essentially a [buffer overflow](https://en.wikipedia.org/wiki/Buffer_overflow)), the stack is said to overflow, typically resulting in a program crash.

5. Heap:

Heap is the segment where dynamic memory allocation usually takes place.

The Heap area is managed by malloc, realloc, and free.

EX: When a memory-layout.c executed memory allocated for data segment, text segment.

