Memory layout of a process | Operating System

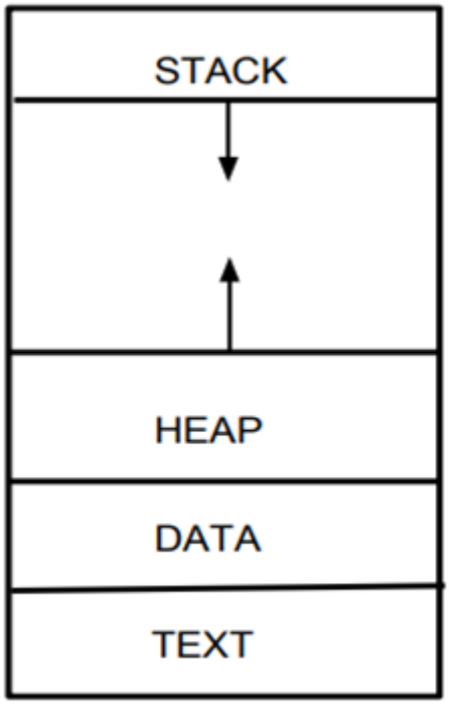
Operating System | Memory layout of a process: Here, we are going to learn about the **memory layout of a process and its sections like: stack, heap, data and text**.
Submitted by Himanshu Singh Bisht, on September 27, 2018

When a C program is created then it is just some pieces of Bytes which is stored in Hard Disk.

When a program is double-clicked in windows then the program starts loading in memory and become a live entity.

The program itself only contains instruction only but a process has more part which are:

1. Stack Section
2. Heap Section
3. Data Section
4. Text Section



1) Stack Section

This section contains local variable, function and returns address. As in the diagram show stack and heap grow in opposite direction which is obvious if both grow in the same direction then they may overlap so it is good if they grow in opposite direction.

Example:

Below function when called will be stored in stack section

```
int fun(int a)
{
        return (a+1);
}
```

As soon as function return the values stack section for that function will get deleted.

2) Heap Section

This Section is used to provide dynamic memory whenever memory is required by the program during runtime It is provided form heap section.

In C language: malloc() and calloc() is used for this purpose

A statement like: malloc(4) will **return the starting address of the 4 BYTE block which is in heap area**.

It is important to note that in C language this dynamic memory needs to be handled i.e. freed when there is no need of it, otherwise heap will become full after some time.

So, free() function is used in the C program to do so.

3) Data Section

This Section contains the global variables and static local variables.

For example in C program

```
#include<stdio.h>
int glbal _var ; // stored in data section
int main()
{
        static int var ;  // stored in data section
        // code statement
        return 0;
}
```

4) Text Section

This section contains the executable instruction, constants, and macros, it is read-only location and is sharable so that can be used by another process also.