Stack Memory Management

BY,

KARTIK SRIVASTAVA

STUDENT ID - 3668516

Topics Covered

- Stacks
- Process Memory
- Stack Memory
- Stack Frames
- Call Stacks

What is a Stack?

- A data structure in which one can add or remove items.
- The item that is added most recently is removed first.
- Stacks follow the “last in, first out” (LIFO) rule
- 3 basic operations are performed in a stack:
  - Push: Adding an element
  - Pop: Removing an element
  - Peek: Returns the element at the top of the stack.

Push and Pop

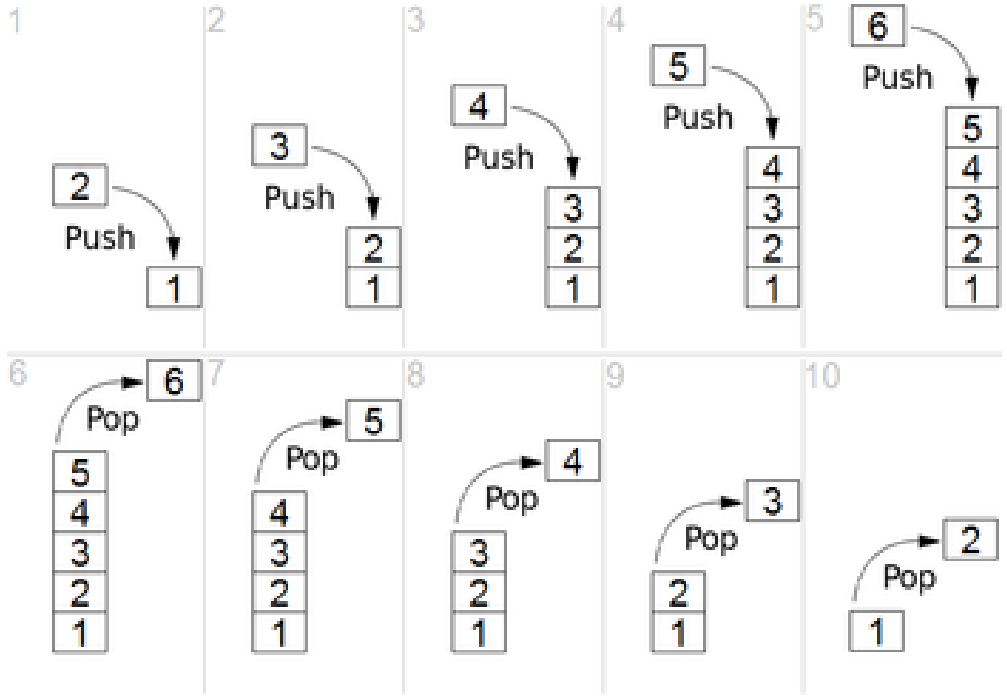
```
case PUSH:
  printf("Value to add: ");
  // Read the element, add it to the stack
  int valE;
  valE = scanf("%d", &val);

  if(valE != 1)
  {
    printf("Invalid input\n");
  }

  if(size < MAX)
  {
      stack[size] = val;
      size++;
  }
```

```
case POP:
// Print out the last element and remove it.
  if(size > 0)
  {
      val = stack[size - 1];
      printf("Removed element: %d\n", val);
      size--;
  }
```

```
C:\Users\srivk>testCode
Choice (1=push, 0=pop, 2=list): 1
Value to add: 11
Choice (1=push, 0=pop, 2=list): 1
Value to add: 23
Choice (1=push, 0=pop, 2=list): 1
Value to add: 35
Choice (1=push, 0=pop, 2=list): 2

11

23

35
Choice (1=push, 0=pop, 2=list): 0
Removed element: 35
Choice (1=push, 0=pop, 2=list): 2

11

23
Choice (1=push, 0=pop, 2=list): 0
Removed element: 23
Choice (1=push, 0=pop, 2=list): 2

11
```

Memory

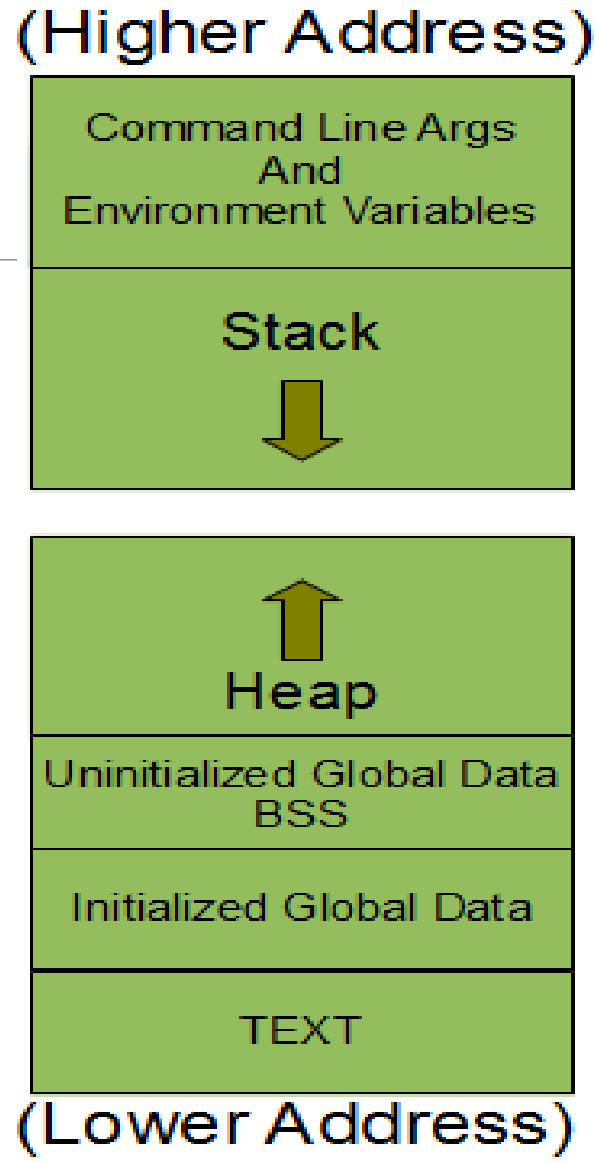
- There are 3 memory types:
  - Stack memory
  - Heap memory
  - Program memory



https://upload.wikimedia.org/wikipedia/commons/thumb/b/b4/Lifo_stack.png/350px-Lifo_stack.png

Process memory

- 4 parts:
  - Text
  - Data
  - Heap
  - Stack
- Addressing starts at the text and increases as the stack approaches.
- The stack is the last region of memory in a process.



https://static.thegeekstuff.com/wp-content/uploads/2012/03/memory-layout.png

Getting the address

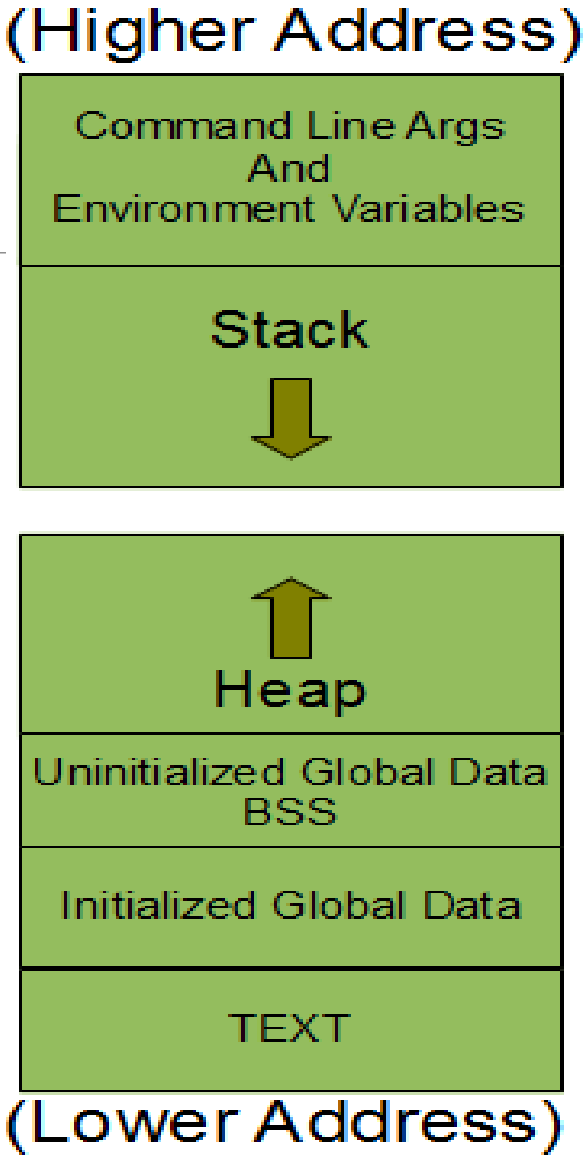
- Using %p

```c
#include <stdio.h>

int main()
{
    int a;
    char b;

    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
}
```

```
C:\Users\srivk>test
Address of a: 0061FF1C
Address of b: 0061FF1B
```

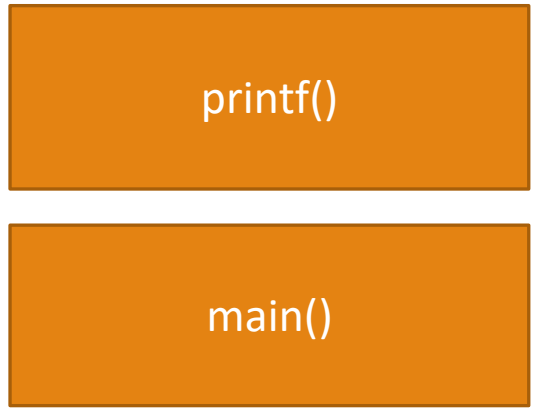
Stack Memory

- Functions are allocated memory on the stack
- Whenever a function is called, a stack frame is generated



Stack Frame

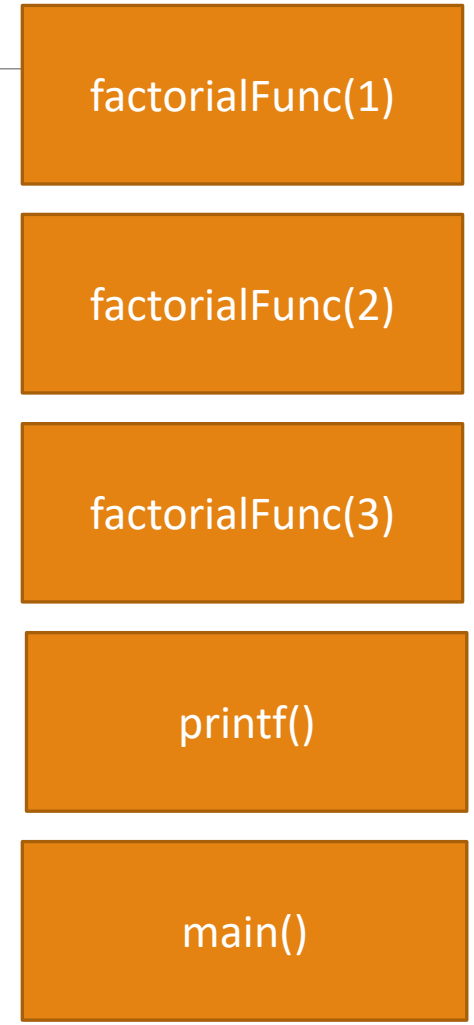
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv)
{
    int num1 = 10;
    int num2 = 11;

    printf("Number 1: %d, Number 2: %d\n", num1, num2);

    return EXIT_SUCCESS;
}
```



Call Stack

- Used by the program to keep a track of function calls
- Composed of stack frames, one for each function call

```c
#include <stdio.h>
#include <stdlib.h>

int factorialFunc(int num)
{
    if(num == 1)
    {
        return 1;
    }
    else
    {
        return num*factorialFunc(num-1);
    }
}

int main()
{
    printf("%i\n", factorialFunc(3));
    return EXIT_SUCCESS;
}
```



Viewing the Call Stack

```c
int main()
{
    int var1;
    int var2;
    int product;

    printf("In main(): ");
    printf("\n");
    printf("Location of var1 in the stack: %p\n", &var1);
    printf("Location of var2 in the stack: %p\n", &var2);
    printf("Location of product in the stack: %p\n", &product);
    printf("\n");

    printf("Enter two numbers: ");
    scanf("%d %d", &var1, &var2);

    product = multiply(var1, var2);
    printf("The product is: %d\n", product);
}
```

```
C:\Users\srivk>test
In main():
Location of var1 in the stack: 0061FF1C
Location of var2 in the stack: 0061FF18
Location of product in the stack: 0061FF14

Enter two numbers: 3
2
In multiply():
Location of num1 in the stack: 0061FF00
Location of num2 in the stack: 0061FF04
Location of product in the stack: 0061FEEC
The product is: 6
```

Thank you!