**Process Memory Map Assignments**

1. What are the various segments of memory created for a process?

A: **Text Segment**: Stores program code.

**Data Segment**: Stores global/static variables (initialized and uninitialized).

**Heap Segment**: Stores dynamically allocated memory.

**Stack Segment**: Stores local variables, function parameters, and return addresses.

**Memory-Mapped Segment**: Used for memory-mapped files or devices (optional).

**Process Control Block**: Stores process information maintained by the OS (not part of the process’s memory layout).

**Kernel Space**: OS memory space used for system-level operations.

**Shared Libraries**: Holds dynamically linked shared libraries used by the process.

2. Refer the code and identify which variable goes in which segment? (Analyse line by line)

#include <stdio.h>

int glob\_array[2000];

void func()

{

static int count = 0;

int func\_local=10;

char \*ptr; /\* Assume pointer size is 4 or 8 bytes. Which segment do these belong to? \*/

char array[1000];

ptr = malloc(100); /\* From which segment, do these 100 bytes come? \*/

count++;

printf("func called %d times\n", count);

}

int main() /\* Which segment do the compiled machine instructions of source code go into? \*/

{

int main\_local;

func();

return 0;

}

glob\_array[2000] -🡪 Data Segment

count 🡪 Data Segment

func\_local 🡪 Stack Segment

ptr 🡪 Stack Segment

array[1000] 🡪 Stack Segment

malloc(100) 🡪 Heap Segment

main\_local 🡪 Stack Segment

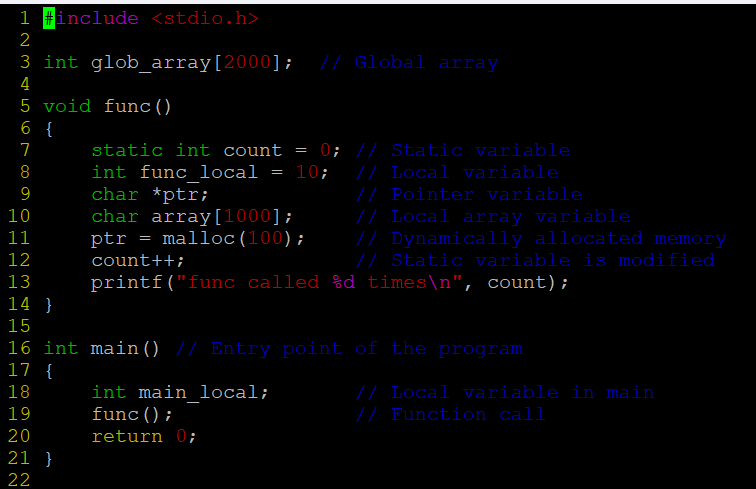
3. How many functions would be in stack when func() is being executed

A: At the point when func() is being executed, there are two functions on the stack:

* 1. The main() function's stack frame (because it is the caller).
  2. The func() function's stack frame (because it is the currently executing function).

Thus, two function call frames will be on the stack when func() is being executed. After func() finishes execution and returns control to main(), the func() stack frame is popped off the stack, and the stack will then only contain main()'s stack frame.

4. Create an executable of above code, run size command and view the size



OUTPUT

