Content 52

Memory Leak In C

#### What is Memory Leak?

A memory leak is a poison for software because software shows undefined behavior due to the memory leak. A memory leak occurs when we create a memory in a heap and forget to delete it. It happens when we do not use dynamic memory properly. If the system has enough memory, in that case also this slows down the performance of the system. To avoid this problem, memory allocated on the heap should always be freed when not needed.

**As an example of a memory leak, let us analyze the following code fragment.**

1. char \*mystring
2. mystring = (char \*)malloc(200);
3. mystring = NULL;

The first line of code char \*mystring declares a pointer to char. In the second line, a space of 200 bytes is reserved. The memory manager returns a pointer to the beginning of that block, and it is stored in the mystring variable. At this point, the address of this block is not stored anywhere else. The following line assigns the value NULL to the same pointer(mystring). Now the question arises, where the address of the memory is reserved? It has been lost, and there is no way to recover it because mystring was the only copy of such value. Consequently, the memory portion will be kept as reserved for the rest of the program execution, so the memory leaked.

#### Reasons for Memory leak:-

Memory leaks are particularly serious issues for programs. Memory leaks in C happen because of these **three reasons**:

* we do not free the memory that is no longer needed
* we do try to free the memory, but we do not have the reference to a dangling pointer.
* we try to free the memory using the wrong function

#### How to avoid memory leak in C?

To avoid memory leaks, we should always be freed the memory allocated on the heap when no longer needed. The code will look like this:

#include <stdlib.h>

void myfunc()

{

int \*ptr = (int \*) malloc(sizeof(int));

/\* Statements \*/

free(ptr);

return;

}

In situations where dynamic allocation is no longer required to maintain, free up space before the program terminates.

**Two rules should be observed for any C program related to dynamic memory management are:**

1. Any memory that is dynamically reserved with malloc, calloc, or realloc must be deallocated through using free().
2. When the program executes its last instruction and has dynamic memory blocks that are not deallocated, it is considered incorrect.

**Code Understanding Memorey Management:**

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char const \*argv[])

{

    int i = 0;

    int \*ptr;

    ptr = (int \*)malloc(50 \* sizeof(int));

    while (i < 100)

    {

        printf("Welcome to VS Code\n");

        if (i % 10 == 0)

        {

            getchar();

        }

        i++;

        free(ptr);      // Without this we will encounter a situation of memory leak

    }

    return 0;

}

**Output:**

Welcome to VS Code

(As I hit Enter)

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Welcome to VS Code

Here If I Again hit Enter It will add 10 lines of “Welcome to VS Code” but First it clear the memory by free(ptr) and continue the next.