


What is Memory Leak? How can we avoid?

Memory leak occurs when programmers create a memory in heap and forget to delete it. Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.




```
/* Function with memory leak */
#include <stdlib.h>

void f()
{
    int *ptr = (int *) malloc(sizeof(int));

    /* Do some work */

    return; /* Return without freeing ptr*/
}
```

To avoid memory leaks, memory allocated on heap should always be freed when no longer needed.



```
/* Function without memory leak */
#include <stdlib.h>;

void f()
{
    int *ptr = (int *) malloc(sizeof(int));

    /* Do some work */

    free(ptr);
    return;
}
```

How to deallocate memory without using free() in C?

Question: How to deallocate dynamically allocate memory without using "free()" function.

Solution: Standard library function `realloc()` can be used to deallocate previously allocated memory. Below is function declaration of "realloc()" from "stdlib.h"

```
void *realloc(void *ptr, size_t size);
```

If "size" is zero, then call to realloc is equivalent to "free(ptr)". And if "ptr" is NULL and size is non-zero then call to realloc is equivalent to "malloc(size)".

```

/* code with memory leak */
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int *ptr = (int*)malloc(10);

    return 0;
}

```

Check the leak summary with valgrind tool. It shows memory leak of 10 bytes, which is highlighted in red colour.

```

[narendra@ubuntu]$ valgrind --leak-check=full ./free
==1238== LEAK SUMMARY:
==1238==      definitely lost: 10 bytes in 1 blocks.
==1238==      possibly lost: 0 bytes in 0 blocks.
==1238==      still reachable: 0 bytes in 0 blocks.
==1238==      suppressed: 0 bytes in 0 blocks.
[narendra@ubuntu]$

```

```

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

int main(void)
{
    int *ptr = (int*) malloc(10);

    /* we are calling realloc with size = 0 */
    realloc(ptr, 0);

    return 0;
}

```

Check the valgrind's output. It shows no memory leaks are possible, highlighted in red color.

```

[narendra@ubuntu]$ valgrind --leak-check=full ./a.out
==1435== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:
==1435== malloc/free: in use at exit: 0 bytes in 0 blocks.
==1435== malloc/free: 1 allocs, 1 frees, 10 bytes allocated.
==1435== For counts of detected errors, rerun with: -v
==1435== All heap blocks were freed -- no leaks are possible.

```