

# Memory Leak and Detection

Yung-Hsiang Lu

# Memory Leak

Memory leak: allocated memory is no longer accessible.

```
int * ip = malloc(16 * sizeof(int));
```



```
/* should free(ip) before malloc again */
```

```
ip = malloc(20 * sizeof(int));
```



```
/* The first array is no longer accessible.
```

```
    The memory is leaked. */
```

# Why is Memory Leak a Problem?

- A program can allocate a finite amount of memory.
- If memory is leaked, eventually malloc reaches the limit and returns NULL.
- Memory leak is a “silent program killer.” The program can run for hours, days, weeks without any problem.
- As more and more memory is leaked, the program becomes slower (due to swapping in virtual memory).
- When malloc finally fails, the program usually crashes.
- Fortunately, tools can be used to detect memory leak.



The image shows a terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help). The code is a C program named "leak.c" that demonstrates memory allocation using malloc. It includes <stdio.h> and <stdlib.h>. The main function takes argc and argv. It declares an integer pointer iptr and a size of 8. It then attempts to allocate memory twice. The first allocation is annotated with a yellow arrow pointing to the malloc call and a yellow box labeled "allocate memory" to its right. The second allocation is also annotated with a yellow arrow pointing to its malloc call. Both allocation attempts include a check for NULL. If either fails, it prints "malloc fail\n" and returns -1. If both succeed, it returns 0. The terminal status bar at the bottom shows "-uu-:---F1 leak.c All (23,1) (C/l Abbrev)-----".

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[])
{
    int * iptr;
    int size = 8;
    /* allocate memory */
    iptr = malloc(size * sizeof(int));
    if (iptr == NULL)
    {
        printf("malloc fail\n");
        return -1;
    }
    /* allocate again */
    iptr = malloc(size * sizeof(int));
    if (iptr == NULL)
    {
        printf("malloc fail\n");
        return -1;
    }
    return 0;
}
```

**allocate memory**

-uu-:---F1 leak.c All (23,1) (C/l Abbrev)-----

[Ubuntu Linux ] valgrind --leak-check=yes ./Leak

==3740== Memcheck, a memory error detector.

==3740== Copyright (C) 200

==3740== Using LibVEX rev

==3740== Copyright (C) 200

==3740== Using valgrind-3.

.

==3740== Copyright (C) 200

==3740== For more details, rerun with: -v

==3740==

==3740==

==3740== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 11 from 1)

==3740== malloc/free: in use at exit: 64 bytes in 2 blocks.

==3740== malloc/free: 2 allocs, 0 frees, 64 bytes allocated.

==3740== For counts of detected errors, rerun with: -v

==3740== searching for pointers to 2 not-freed blocks.

==3740== checked 51,084 bytes.

==3740==

==3740== 32 bytes in 1 blocks are definitely lost in loss record 1 of 2

==3740== at 0x4026FDE: malloc (vg\_replace\_malloc.c:207)

==3740== by 0x8048445: main (leak.c:16)

==3740==

==3740==

==3740== 32 bytes in 1 blocks are definitely lost in loss record 2 of 2

==3740== at 0x4026FDE: malloc (vg\_replace\_malloc.c:207)

==3740== by 0x8048419: main (leak.c:9)

==3740==

==3740== LEAK SUMMARY:

==3740== definitely lost: 64 bytes in 2 blocks.

==3740== possibly lost: 0 bytes in 0 blocks.

command to check memory leak

**valgrind --leak-check=yes**

leak  $4 \times 8 \times 2 = 64$  bytes

Valgrind Home - Mozilla Firefox

File Edit View History Bookmarks Tools Help

< >

↺

✕

🏠

 http://valgrind.org/

☆

 Google



Information

About

News

Tool Suite

Supported Platforms

The Developers

Source Code

Current Releases

Release Archive

Front Ends / GUIs

Variants / Patches

Code Repository

Documentation

Table of Contents

Quick Start

FAQ

User Manual

Download Manual

Research Papers

Books

Contact

Mailing Lists

# Valgrind



Current release: [valgrind-3.5.0](#)

Valgrind is an award-winning instrumentation framework for building dynamic analysis tools. There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile your programs in detail. You can also use Valgrind to build new tools.

The Valgrind distribution currently includes six production-quality tools: a

Done