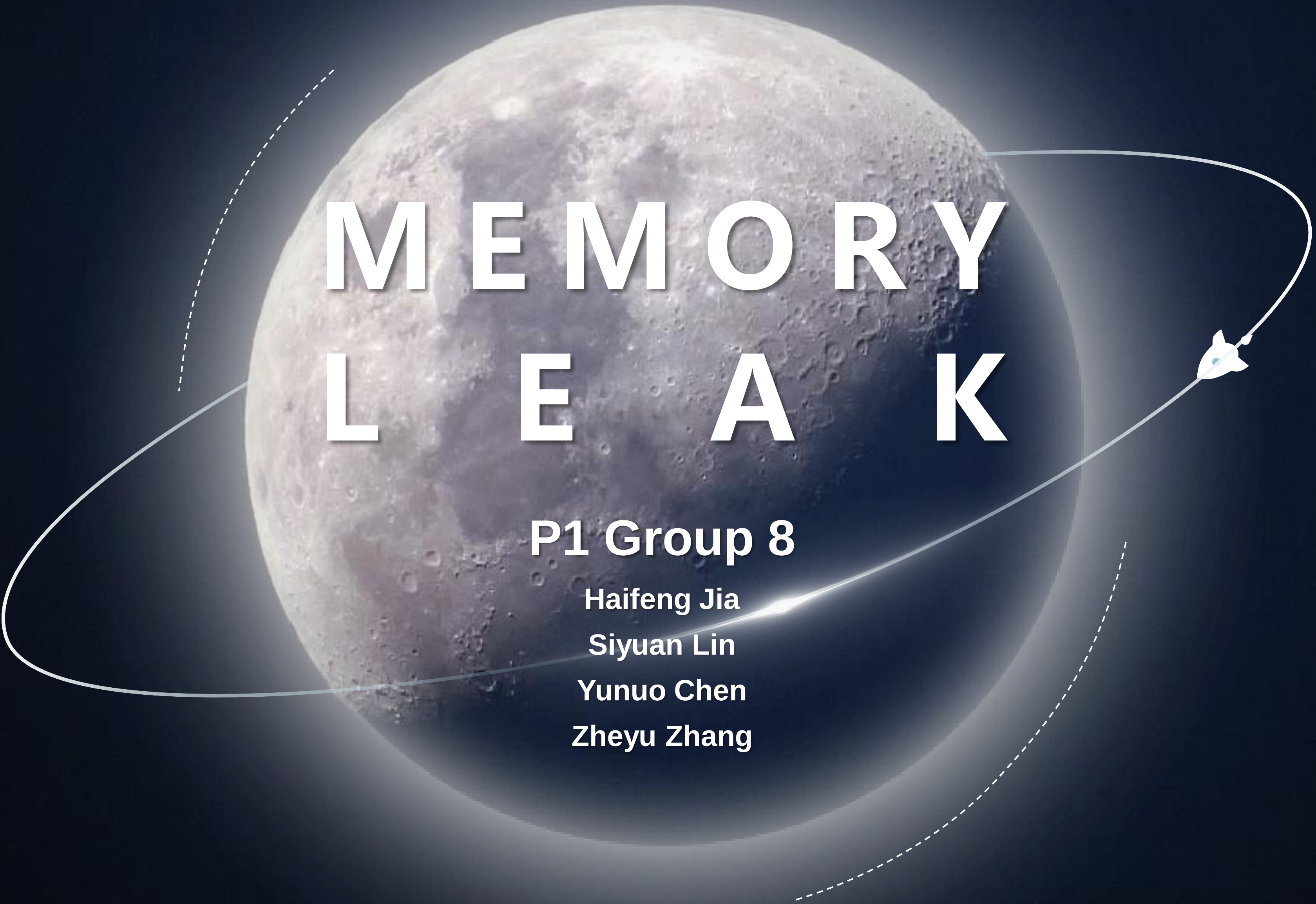


MEMORY LEAK

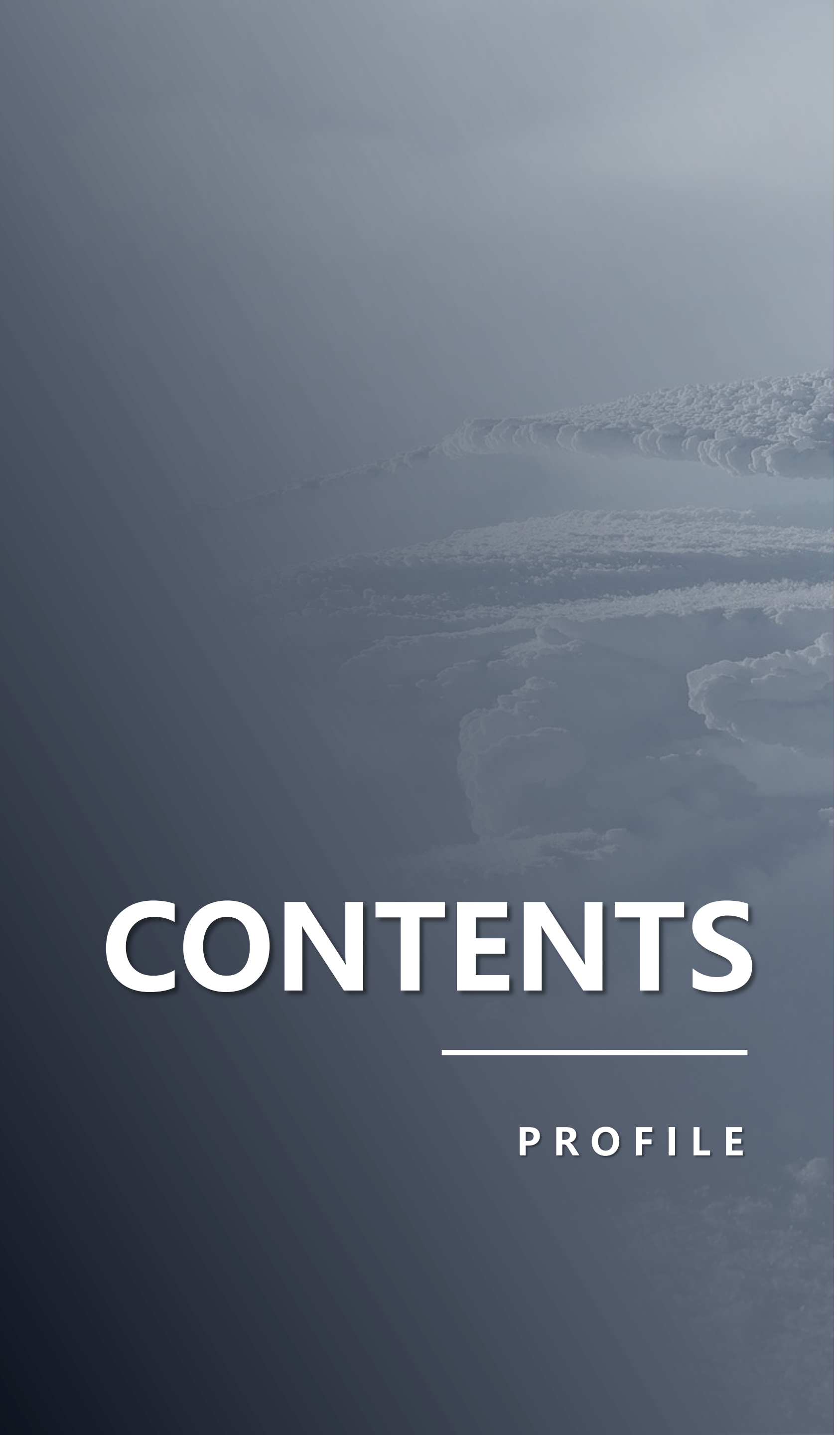
P1 Group 8

Haifeng Jia

Siyuan Lin

Yunuo Chen

Zheyu Zhang

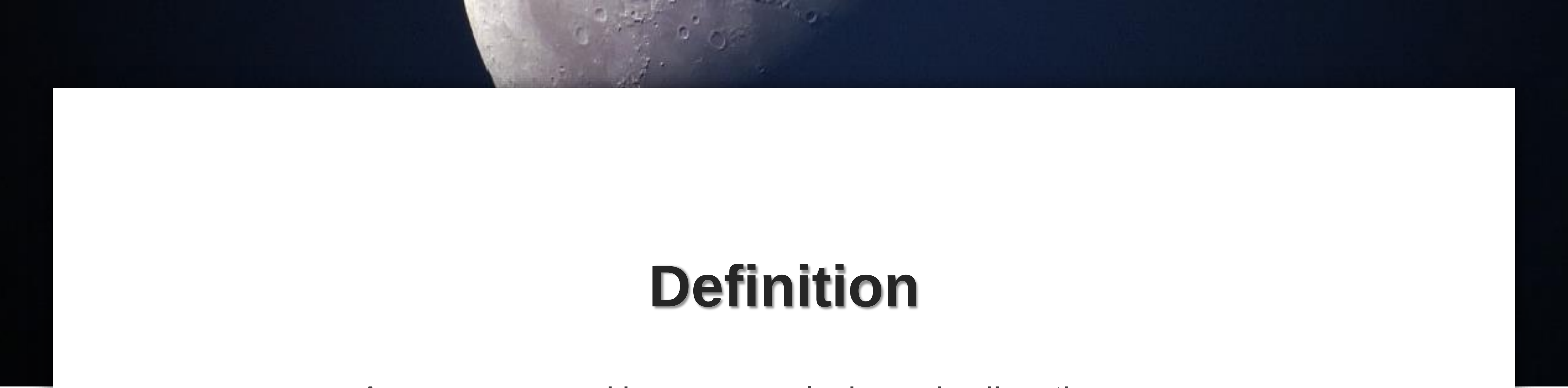


CONTENTS

PROFILE

01

Definition



Definition

- An error occurred in a program’s dynamic allocation process that parts of memory become unusable or hidden.
- It leads to resident program leak additional memory space without releasing previous space.
- It will lead to the exhaustion of memory resources and poorly performing or frozen system.

Definition

A memory leak is a particular type of unintentional memory consumption by a computer program where the program fails to release memory when no longer needed. This condition is normally the result of a bug in a program that prevents it from freeing up memory that it no longer needs. This term has the potential to be confusing, since memory is not physically lost from the computer. Rather, memory is allocated to a program, and that program subsequently loses the ability to access it due to program logic flaws.

02

Types and Reasons

Types

- **Leaked temporary work space.**

  e.g Inside a function

- **Leaked data member.**

  Not freed before class is destroyed

- **Leaked global member.**

  Used multiple times but not free

- **Calling the wrong destructor.**

Reasons

- **Forget to release / free the resource**

  Memory, close file…

- **Double free**

  A child process frees again or loop continuous free

- **Base class doesn't have a virtual destructor and a base pointer points to the inherited class**

Types & Reasons

For C or C++ programs without Garbage Collection,

we mainly divide memory leak into 2 types:

Heap leak

Memory refers to a chunk of memory allocated from the heap manually during the execution of the program, and then must be deleted by calling the corresponding free or delete.

Resource Leak

Resource leak refers to a program that uses resources allocated by the system, such as Bitmap, Handle, and SOCKET, but does not use corresponding functions to release them.

Types & Reasons

Reminder:

Linux also produces memory leaks when deadlocks occur (deadlocks: threads A and B are holding the lock and requesting the lock at the same time).

03

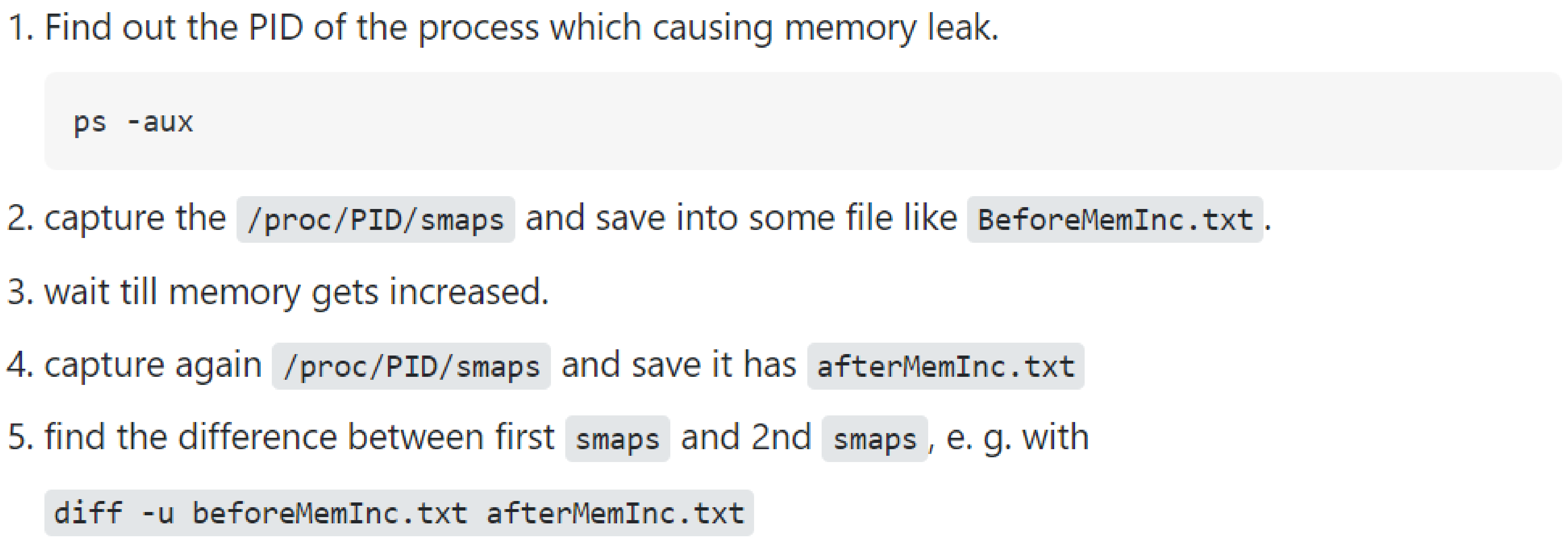
Memory Analysis & Check

I. How to check memory leak before the start of a process?

1. Use `valgrind --leak-check=full ./*` to run the program

2. Add `-fsanitize=leak -fsanitize=address` when compiling the program.

II. How to find a memory leak of a running process?

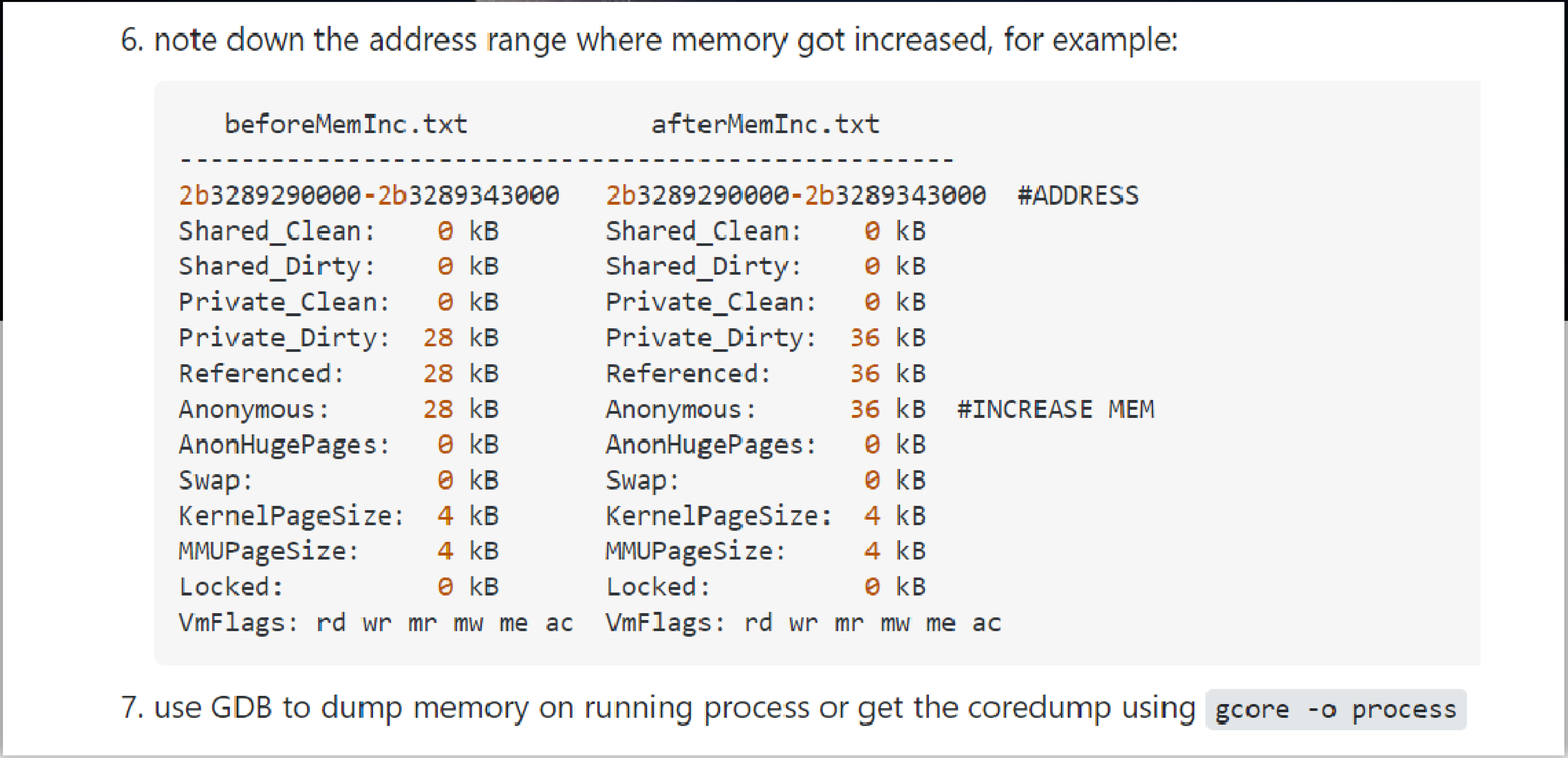


1. Find out the PID of the process which causing memory leak.

```
ps -aux
```

2. capture the `/proc/PID/smaps` and save into some file like `BeforeMemInc.txt`.
3. wait till memory gets increased.
4. capture again `/proc/PID/smaps` and save it has `afterMemInc.txt`
5. find the difference between first `smaps` and 2nd `smaps`, e. g. with

   `diff -u beforeMemInc.txt afterMemInc.txt`



6. note down the address range where memory got increased, for example:

```
   beforeMemInc.txt            afterMemInc.txt
---------------------------------------------------
2b3289290000-2b3289343000   2b3289290000-2b3289343000  #ADDRESS
Shared_Clean:    0 kB       Shared_Clean:    0 kB
Shared_Dirty:    0 kB       Shared_Dirty:    0 kB
Private_Clean:   0 kB       Private_Clean:   0 kB
Private_Dirty:  28 kB       Private_Dirty:  36 kB
Referenced:     28 kB       Referenced:     36 kB
Anonymous:      28 kB       Anonymous:      36 kB  #INCREASE MEM
AnonHugePages:   0 kB       AnonHugePages:   0 kB
Swap:            0 kB       Swap:            0 kB
KernelPageSize:  4 kB       KernelPageSize:  4 kB
MMUPageSize:     4 kB       MMUPageSize:     4 kB
Locked:          0 kB       Locked:          0 kB
VmFlags: rd wr mr mw me ac  VmFlags: rd wr mr mw me ac
```

7. use GDB to dump memory on running process or get the coredump using `gcore -o process`

8. I used gdb on running process to dump the memory to some file.

```
gdb -p PID
dump memory ./dump_outputfile.dump 0x2b3289290000 0x2b3289343000
```

9. now, use `strings` command or `hexdump -C` to print the `dump_outputfile.dump`

```
strings outputfile.dump
```

10. You get readable form where you can locate those strings into your source code.
11. Analyze your source to find the leak.

III. Other tools for C/C++ programs

- **ccmalloc**: A simple memory leak and MALloc debugging library for C and C++ programs on Linux and Solaris
- **LeakTracer**: traces and analyzes memory leaks in C++ programs on Linux, Solaris, and HP-UX
- **Electric Fence**: A debugging library for Malloc () written by Bruce Perens in the Linux

III. Other tools for C/C++ programs

- **Leaky**: A program that detects memory leaks in Linux
- **Dmalloc**: Debug Malloc Library
- **MEMWATCH**: Written by Johan Lindh, MEMWATCH is an open source C language memory error detection tool, mainly through GCC's Processor.
- **KCachegrind**: A visualization tool for the profiling data generated by Cachegrind and Calltree

IV. Linux kernel memory leak detection tool: Kmemleak

Kmemleak provides a way of detecting possible kernel memory leaks in a way similar to a tracing garbage collector (https://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29#Tracing_garbage_collectors), with the difference that the orphan objects are not freed but only reported via /sys/kernel/debug/kmemleak.

04

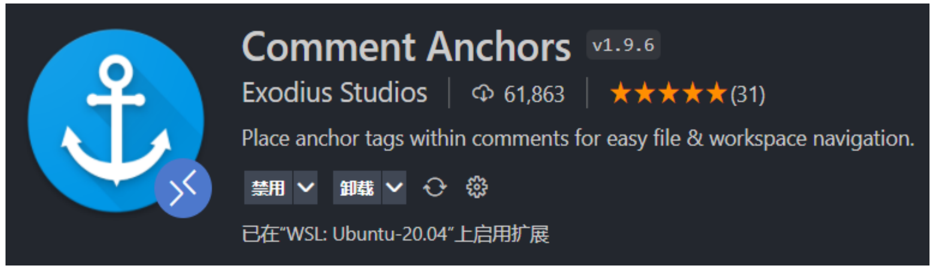
Tips & Solutions

Tips & Solutions

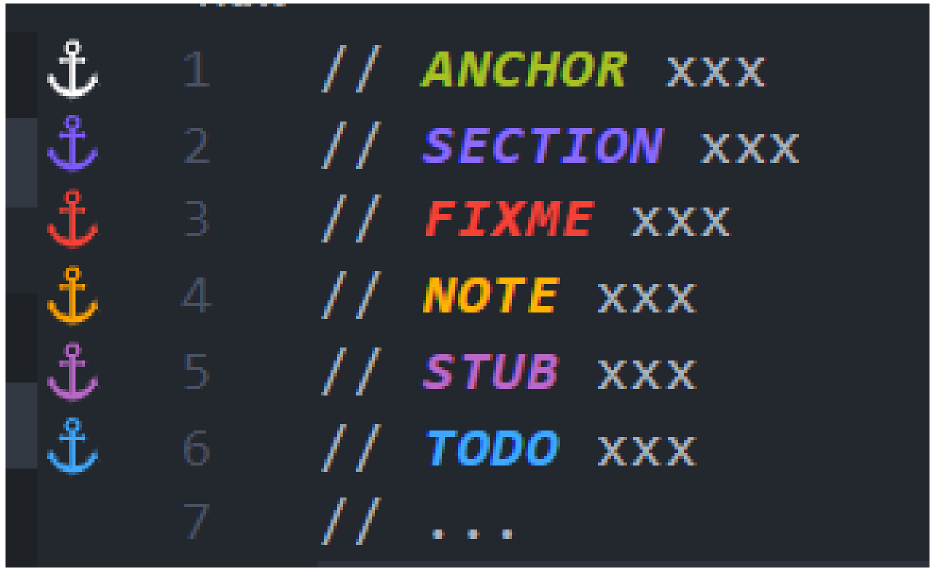
1. Develop good programming habits
2. When applying for memory, remember to release
3. Remember to always use third-party tools to detect memory leaks
4. Use smart pointers
5. Avoid deadlocks
6. Close all the opened file descriptors

Tips & Solutions

VS Code extension

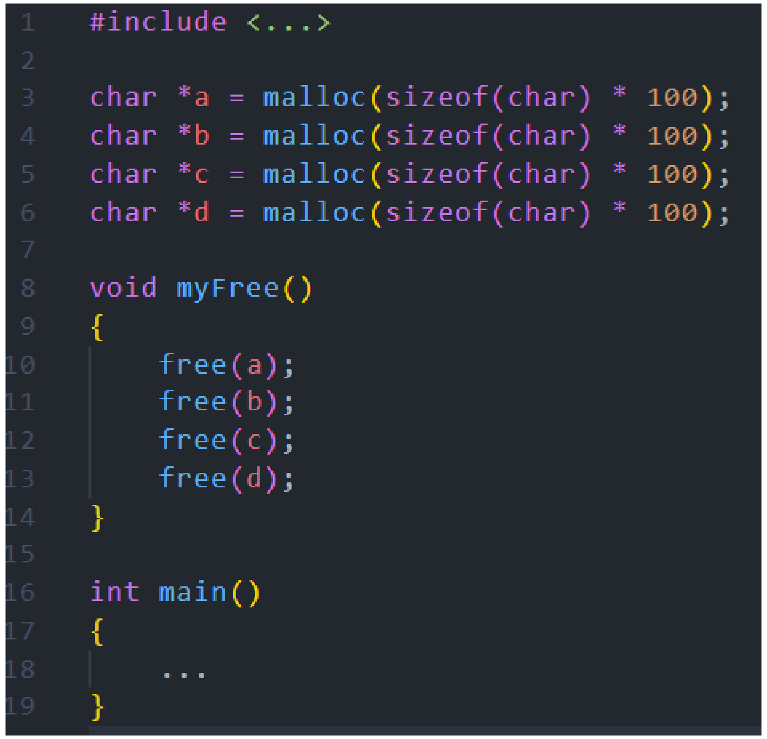

Comment Anchors v1.9.6
Exodius Studios | 61,863 | ★★★★★ (31)
Place anchor tags within comments for easy file & workspace navigation.

已在"WSL: Ubuntu-20.04"上启用扩展


```
// ANCHOR xxx
// SECTION xxx
// FIXME xxx
// NOTE xxx
// STUB xxx
// TODO xxx
// ...
```

Tips & Solutions

Global Variables & Grouped Free



```c
#include <...>

char *a = malloc(sizeof(char) * 100);
char *b = malloc(sizeof(char) * 100);
char *c = malloc(sizeof(char) * 100);
char *d = malloc(sizeof(char) * 100);

void myFree()
{
    free(a);
    free(b);
    free(c);
    free(d);
}

int main()
{
    ...
}
```

Reference Page

[1] https://www.golinuxcloud.com/how-to-find-memory-leaks/#Conclusion

[2] https://www.kernel.org/doc/html/v5.0/dev-tools/kmemleak.html

[3] https://www.golinuxcloud.com/tutorial-linux-memory-management-overview/

[4] https://unix.stackexchange.com/questions/36450/how-can-i-find-a-memory-leak-of-a-running-process

[5] https://www.golinuxcloud.com/how-to-find-memory-leaks/

THANKS