MEMORY LEAKS DETECTION

Joel Romero Botella

Project II
UPC-CITM

INDEX

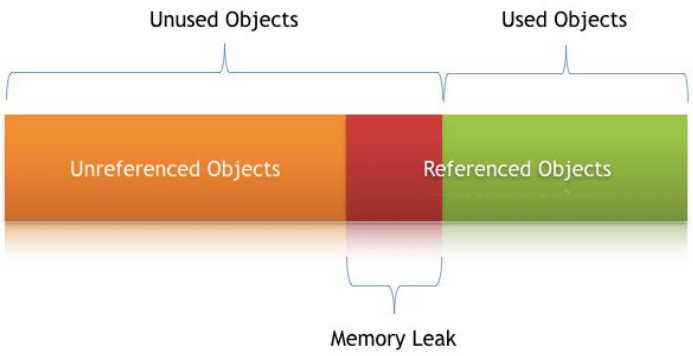
1

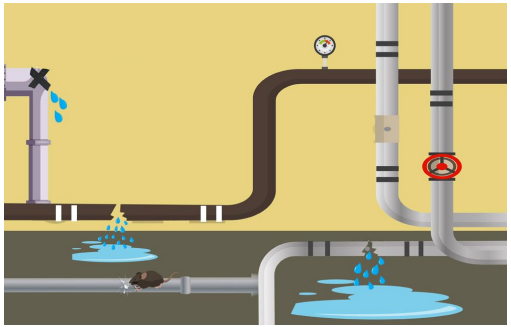
WHAT ARE MEMORY LEAKS?

Let’s start with an introduction

What are Memory Leaks?

Memory leaks are a type of programming error where memory that is no longer needed by a program is not properly released or freed, causing the program to continue using more and more memory over time.





What are Memory Leaks?

This is an example of what would happen if there is a memory leak present in our video game:

What are Memory Leaks?

The best case:

7000 MB RAM

12000 MB RAM

What are Memory Leaks?

The worst case:

7000 MB RAM

GOODBYE!!!

2

CAUSES OF MEMORY LEAKS

How many types of causes

Causes of Memory Leaks

- Efficiency reduction
- Stability Issues
- Errors and unexpected behavior
- Save Issues
- Debugging difficulties

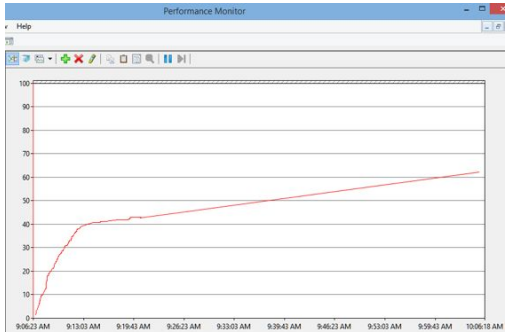
3

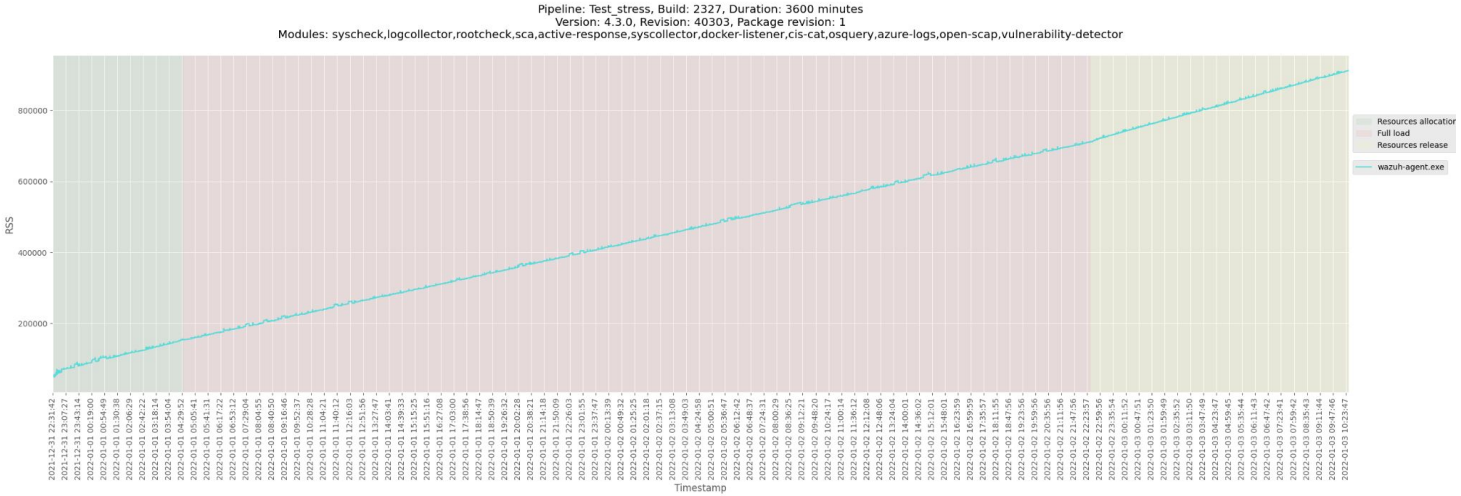
DETECTION

How to detect the Memory Leaks and solve them

Detection

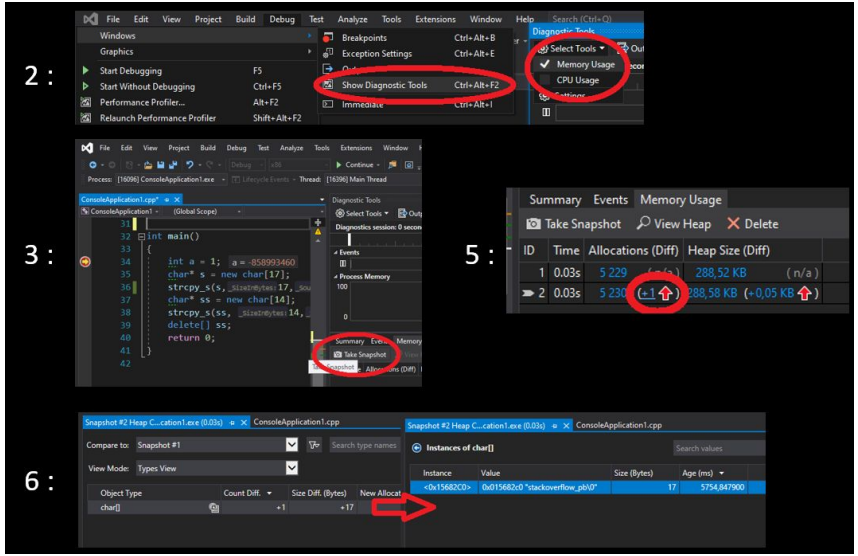
Stress test

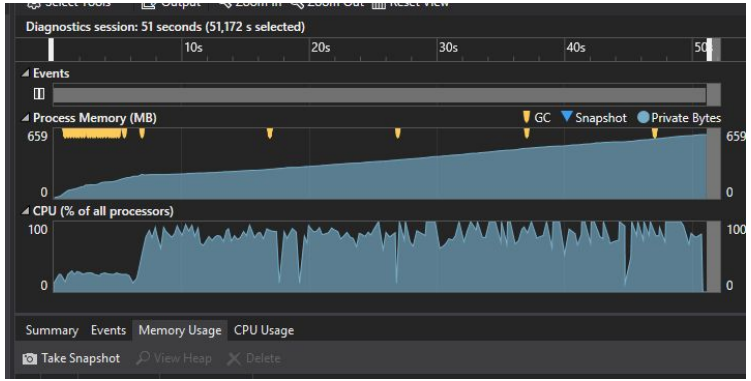




Detection

Memory profiling

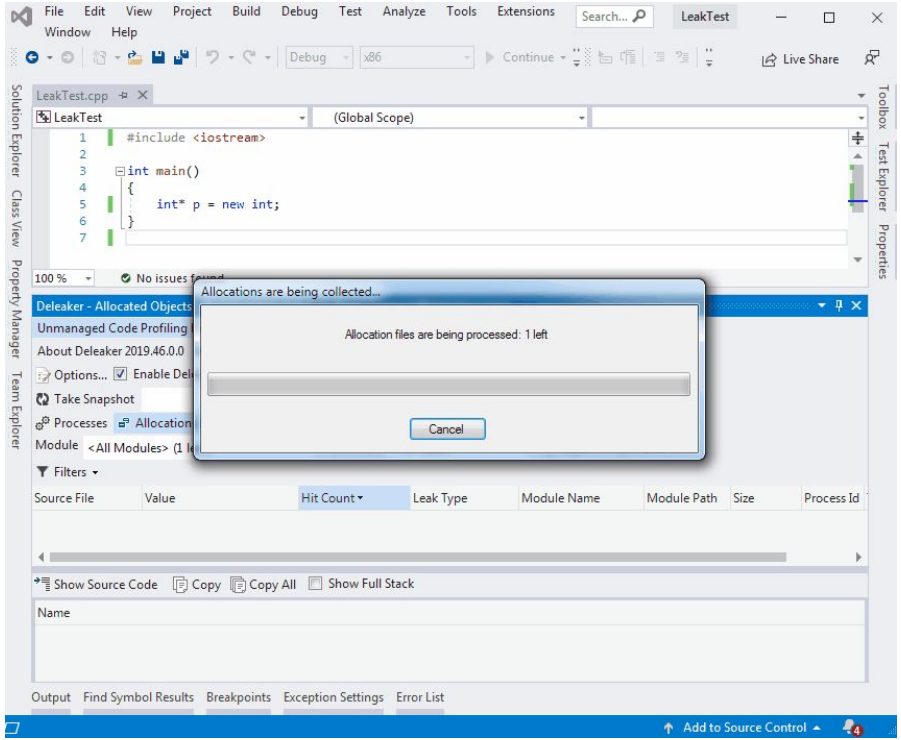




Detection

Memory analysis tools

```
tutorialadda@tutorialadda:~/valgrind$
tutorialadda@tutorialadda:~/valgrind$ gcc -o test test.c -g
tutorialadda@tutorialadda:~/valgrind$
tutorialadda@tutorialadda:~/valgrind$
tutorialadda@tutorialadda:~/valgrind$
tutorialadda@tutorialadda:~/valgrind$
tutorialadda@tutorialadda:~/valgrind$ valgrind --tool=memcheck --leak-check=yes  ./test
==24562== Memcheck, a memory error detector
==24562== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==24562== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==24562== Command: ./test
==24562==
==24562==
==24562== HEAP SUMMARY:
==24562==     in use at exit: 10 bytes in 1 blocks
==24562==   total heap usage: 2 allocs, 1 frees, 25 bytes allocated
==24562==
==24562== 10 bytes in 1 blocks are definitely lost in loss record 1 of 1
==24562==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==24562==    by 0x400577: main (test.c:8)
==24562==
==24562== LEAK SUMMARY:
==24562==    definitely lost: 10 bytes in 1 blocks
==24562==    indirectly lost: 0 bytes in 0 blocks
==24562==      possibly lost: 0 bytes in 0 blocks
==24562==    still reachable: 0 bytes in 0 blocks
==24562==         suppressed: 0 bytes in 0 blocks
==24562==
==24562== For counts of detected and suppressed errors, rerun with: -v
==24562== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
tutorialadda@tutorialadda:~/valgrind$
```



Detection

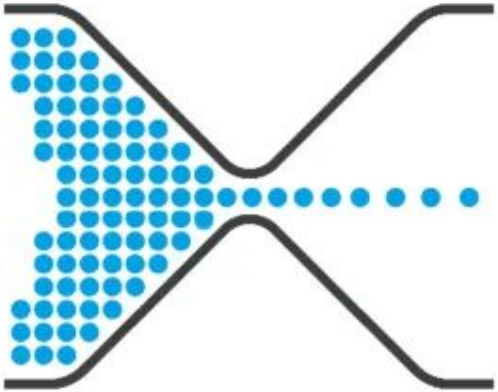
Manual tests

- Source code analysis
- Analysis of debug logs

Detection

Release memory

- Use functions like “free” (C, C++)
- If you don’t release…



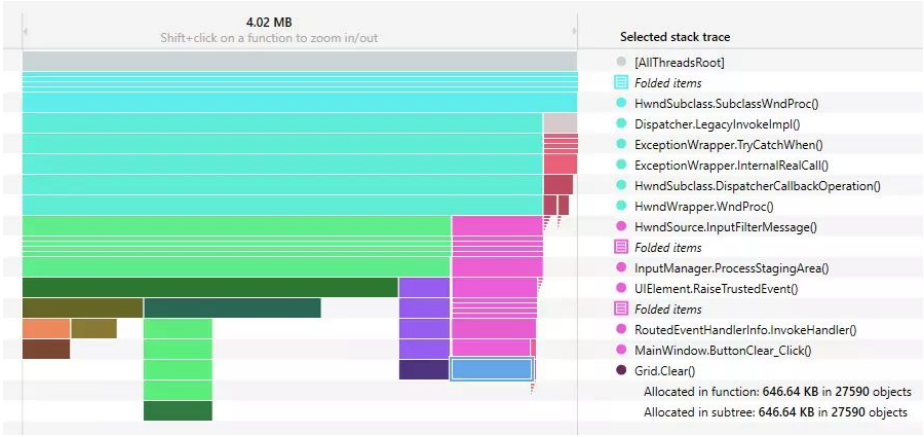
Bottleneck effect

4

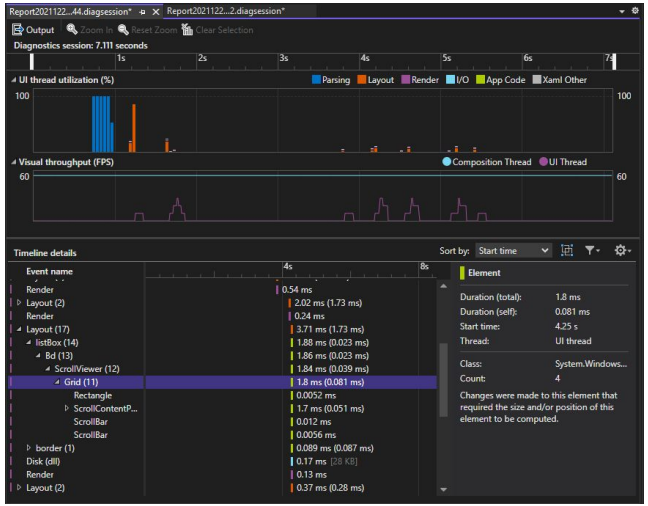
SOFTWARES

How many softwares are there?

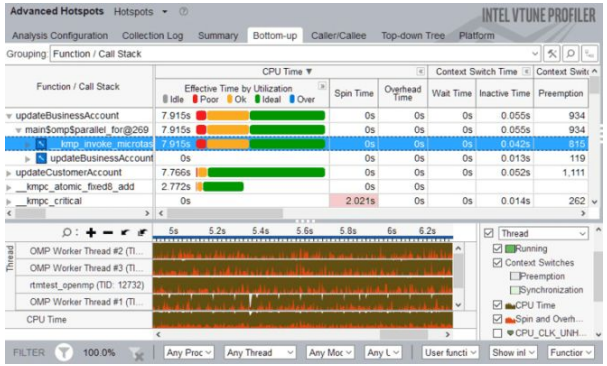
Softwares



DotMemory



Visual Studio Memory Profiler



Intel VTune Amplifier

Softwares

Valgrind

```
nikesh@poison: ~
File  Edit  View  Search  Terminal  Help
--1856-- REDIR: 0x40aeef0 (malloc) redirected to 0x40267df (malloc)
--1856-- REDIR: 0x40b57e0 (strchrnul) redirected to 0x4028b3c (strchrnul)
hello valgrid!!--1856-- REDIR: 0x40af3b0 (free) redirected to 0x4025b6b (free)
==1856==
==1856== HEAP SUMMARY:
==1856==     in use at exit: 100 bytes in 1 blocks
==1856==   total heap usage: 1 allocs, 0 frees, 100 bytes allocated
==1856==
==1856== Searching for pointers to 1 not-freed blocks
==1856== Checked 56,628 bytes
==1856==
==1856== 100 bytes in 1 blocks are definitely lost in loss record 1 of 1
==1856==    at 0x4026864: malloc (vg_replace_malloc.c:236)
==1856==    by 0x8048408: main (test.c:6)
==1856==
==1856== LEAK SUMMARY:
==1856==    definitely lost: 100 bytes in 1 blocks
==1856==    indirectly lost: 0 bytes in 0 blocks
==1856==      possibly lost: 0 bytes in 0 blocks
==1856==    still reachable: 0 bytes in 0 blocks
==1856==         suppressed: 0 bytes in 0 blocks
==1856==
==1856== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 11 from 6)
--1856--
--1856-- used_suppression:     11 U1004-ARM-_dl_relocate_object
==1856==
==1856== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 11 from 6)
nikesh@poison:~$
```

How to use Valgrind

1. Install Valgrind

Recommend to use Linux if you have.

Execute the next command at the Linux terminal:

sudo apt-get install valgrind

How to use Valgrind

2. Compile the code with depuration mode

This allows Valgrind to access additional information about memory usage in the program.

Execute the next command at the Linux terminal:

```
swdev@silver-pc:~/Desktop/MemoryLeak$ gcc -g progErrors.c -o myprog
```

How to use Valgrind

3. Run Valgrind

The next command can analyze if the code has memory leaks or not.

Execute the next command at the Linux terminal:

```
swdev@silver-pc:~/Desktop/MemoryLeak$ valgrind --leak-check=yes ./mypro
g
```

How to use Valgrind

4. Analyze Valgrind report

Summary:

```
HEAP SUMMARY:
    in use at exit: 10,000 bytes in 100 blocks
  total heap usage: 100 allocs, 0 frees, 10,000 bytes allocate
```

Error:

```
==7308== 9,900 bytes in 99 blocks are definitely lost in loss record 2
of 2
==7308==    at 0x4C2DBF6: malloc (vg_replace_malloc.c:299)
==7308==    by 0x400540: main (progErrors.c:10)
```

How to use Valgrind

4. Analyze Valgrind report

Leak Summary:

```
==7308== LEAK SUMMARY:
==7308==    definitely lost: 9,900 bytes in 99 blocks
==7308==    indirectly lost: 0 bytes in 0 blocks
==7308==      possibly lost: 0 bytes in 0 blocks
==7308==    still reachable: 100 bytes in 1 blocks
==7308==         suppressed: 0 bytes in 0 blocks
==7308== Reachable blocks (those to which a pointer was found) are not
shown.
```

How to use Valgrind

5. Fix memory leaks

Did you see the problem??

```c
#include <stdlib.h>
#include <unistd.h>

char *ptr = NULL;

int main(void)
{
    for(int i = 0; i < 100; i++)
    {
        ptr = malloc(100);
    }

    return 0;
}
```

How to use Valgrind

6. Run Valgrind again

When you solve the problem, try again to check it.

5

IMPLEMENTATION IN PROJECT II

What will we use to detect memory leaks?

Implementation in Project II

- What software will we use?
- Will we use only software or also leak analysis with other techniques mentioned above?
- What is better?

6

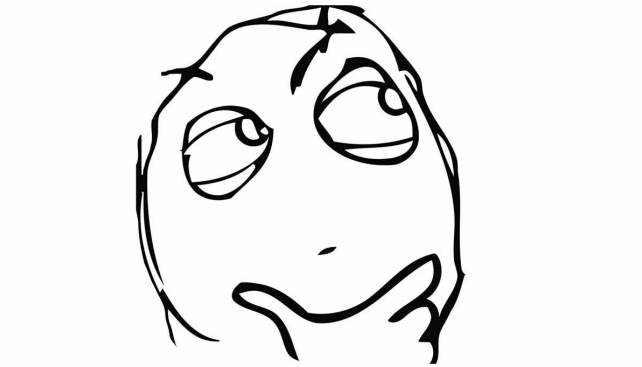
MANUAL TECHNIQUES OR SOFTWARE?

Let's get into the discussion

Implementation in Project II

I want you to have a short debate about which is better.

Manual techniques or software?



7

CONCLUSION

We are already finishing the presentation

Conclusion

Manual	Software
Simpler and more accessible	Automate the process
Better understanding of the code	Provide detailed information
Flexible and adaptable	Detect complex problems
More prone to human error	Complicated to use
Slower and more time to fix it	Can cost a lot of money

THANKS !!!

I hope that this presentation will be useful to you

References

This presentation has been referenced from the following web pages and videos:

https://en.wikipedia.org/wiki/Memory_leak

https://www.it.uc3m.es/pbasanta/asng/course_notes/memory_profiler_es.html#:~:text=Valgrind%20es%20un%20sistema%20de,los%20programas%20sean%20m%C3%A1s%20estables.

https://valgrind.org/

https://www.makeuseof.com/what-is-a-memory-leak/

https://www.youtube.com/watch?v=NMmK8o_BZ7M&ab_channel=WhileTrueThenDream