Chapter 7:
LeakSanitizer Developed by Google

Mijin An
meeeeeejin@gmail.com

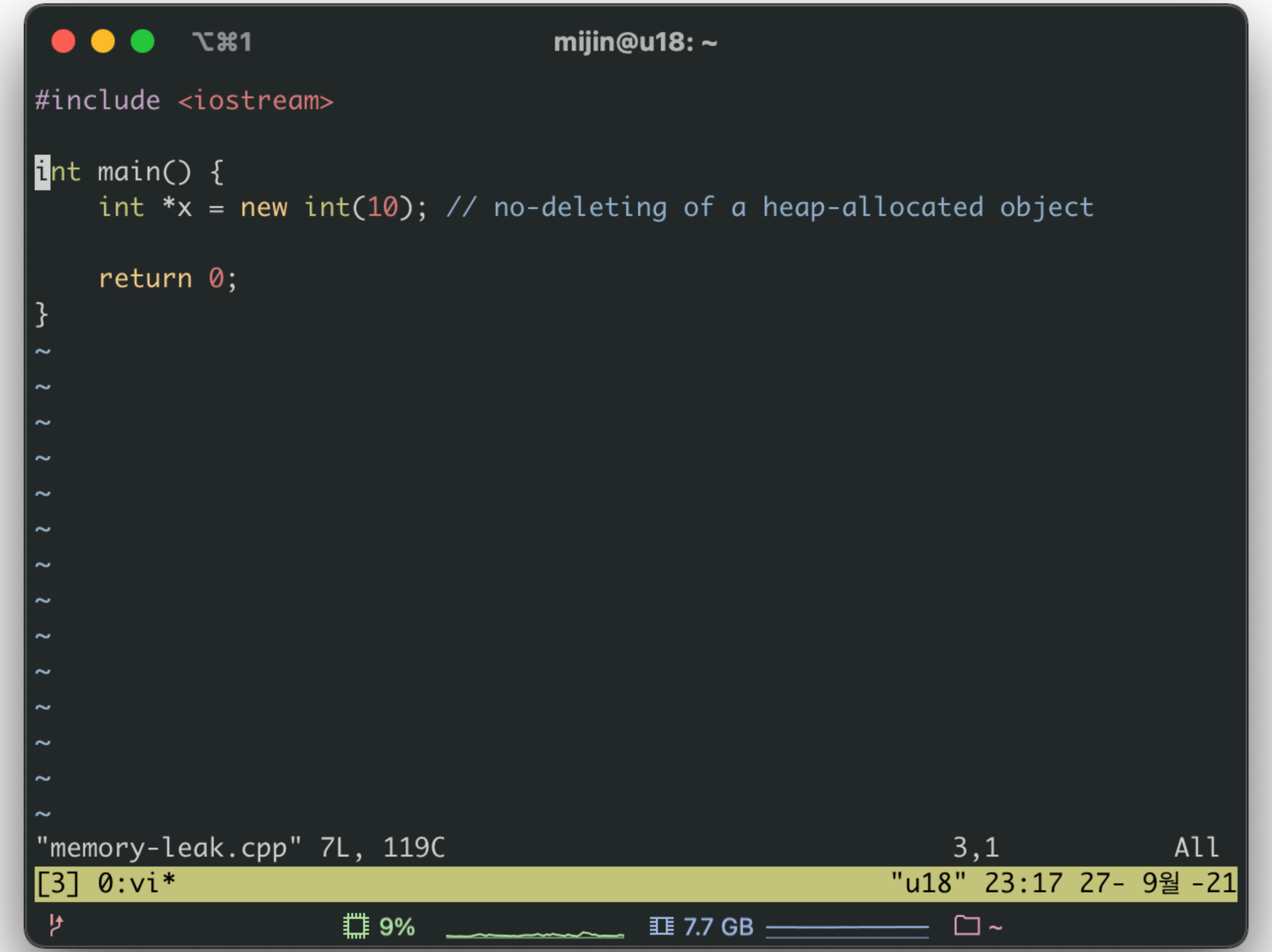
LeakSanitizer

- A **run-time memory leak detector** developed by Google
  - Supported platforms: x86_64 Linux and OS X

- It can be combined with **AddressSanitizer** to get both memory error and leak detection

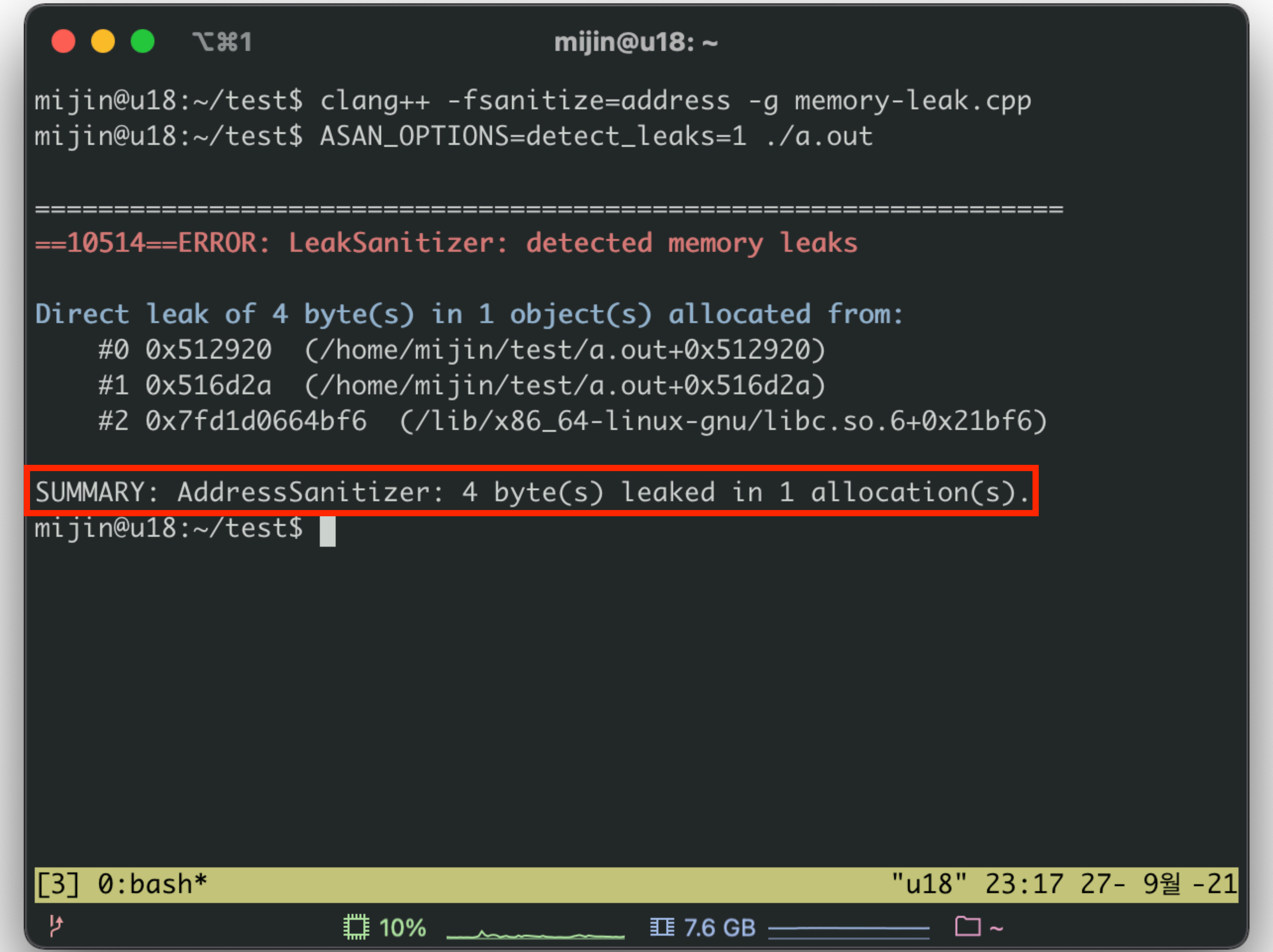
- LeakSanitizer adds almost **no performance overhead** until the very end of the process, at which point there is an extra leak detection phase
  - **Memory leak detection is only conducted before exiting the program**

Example on Memory Leaks in C++: Code



```
#include <iostream>

int main() {
    int *x = new int(10); // no-deleting of a heap-allocated object

    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"memory-leak.cpp" 7L, 119C                              3,1           All
[3] 0:vi*                                          "u18" 23:17 27- 9월 -21
```

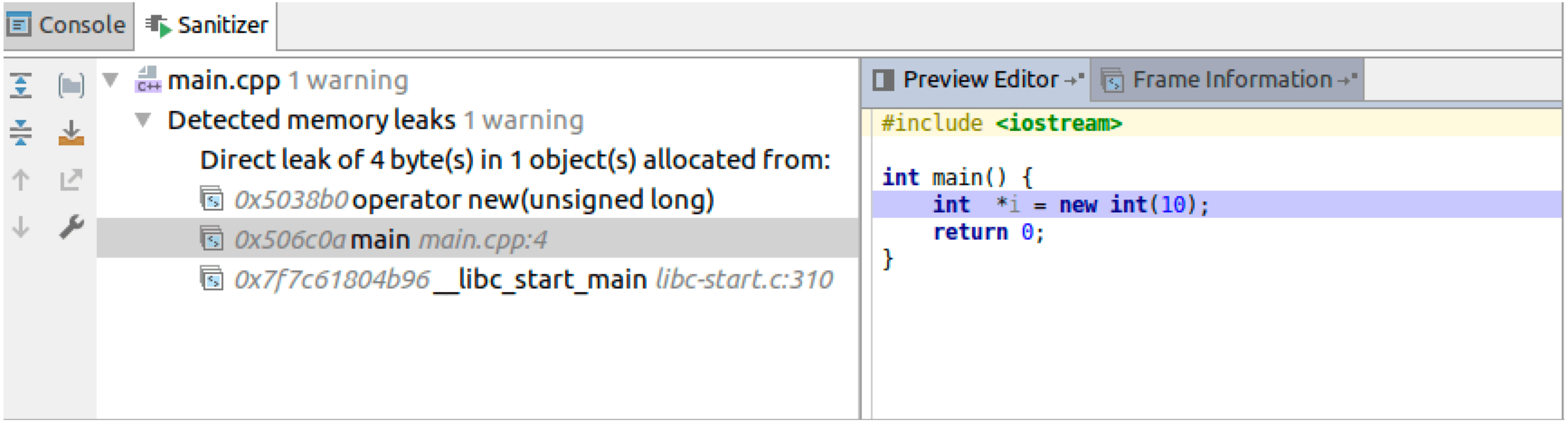
Example on Memory Leaks in C++: Result



```
mijin@u18:~/test$ clang++ -fsanitize=address -g memory-leak.cpp
mijin@u18:~/test$ ASAN_OPTIONS=detect_leaks=1 ./a.out

=================================================================
==10514==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 4 byte(s) in 1 object(s) allocated from:
    #0 0x512920  (/home/mijin/test/a.out+0x512920)
    #1 0x516d2a  (/home/mijin/test/a.out+0x516d2a)
    #2 0x7fd1d0664bf6  (/lib/x86_64-linux-gnu/libc.so.6+0x21bf6)

SUMMARY: AddressSanitizer: 4 byte(s) leaked in 1 allocation(s).
mijin@u18:~/test$
```

```
[3] 0:bash*                                              "u18" 23:17 27- 9월 -21
```

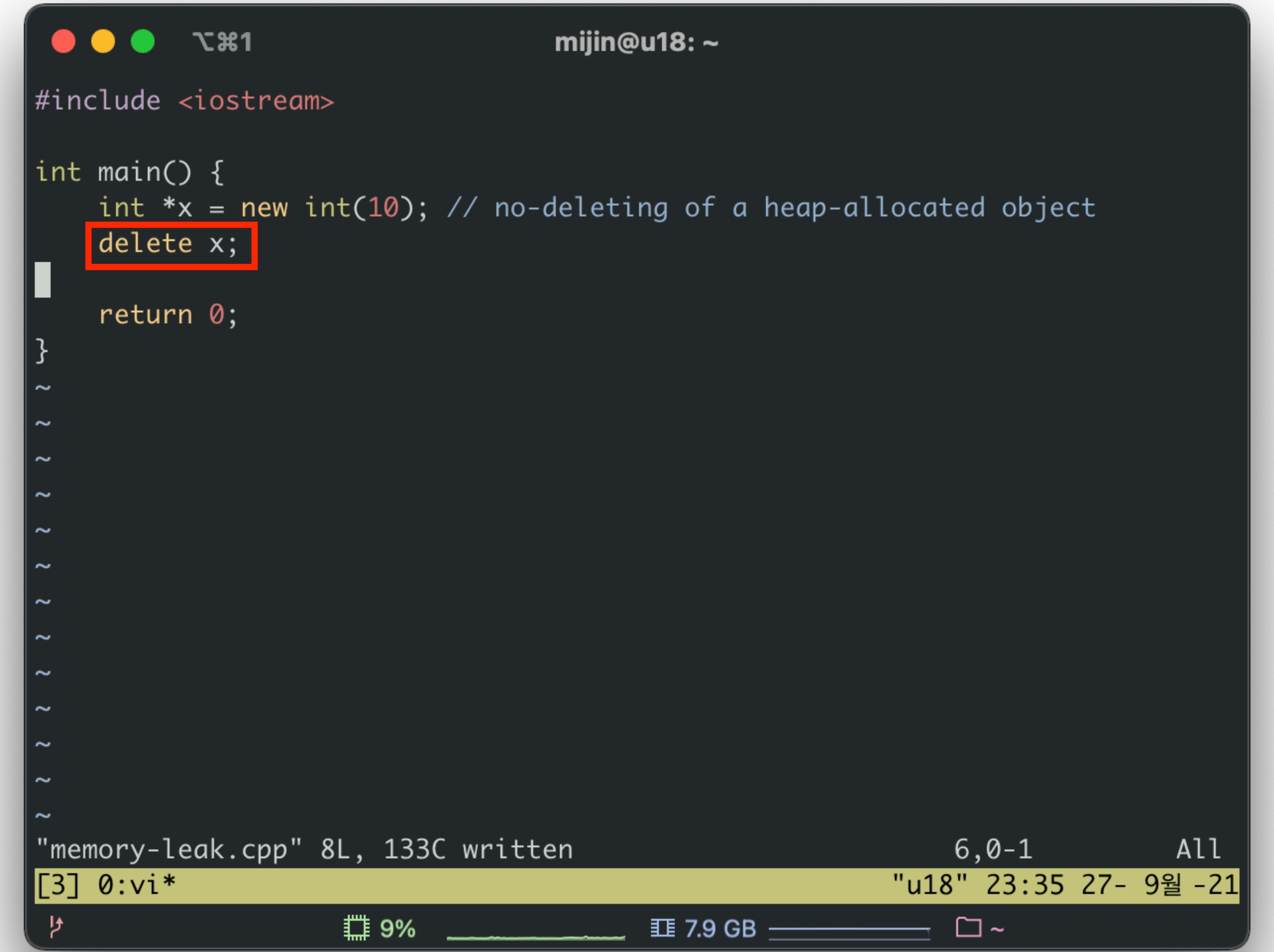
Example on Memory Leaks with CLion



```
main.cpp 1 warning
  Detected memory leaks 1 warning
    Direct leak of 4 byte(s) in 1 object(s) allocated from:
      0x5038b0 operator new(unsigned long)
      0x506c0a main main.cpp:4
      0x7f7c61804b96 __libc_start_main libc-start.c:310
```

```cpp
#include <iostream>

int main() {
    int  *i = new int(10);
    return 0;
}
```

Fix Memory Leaks

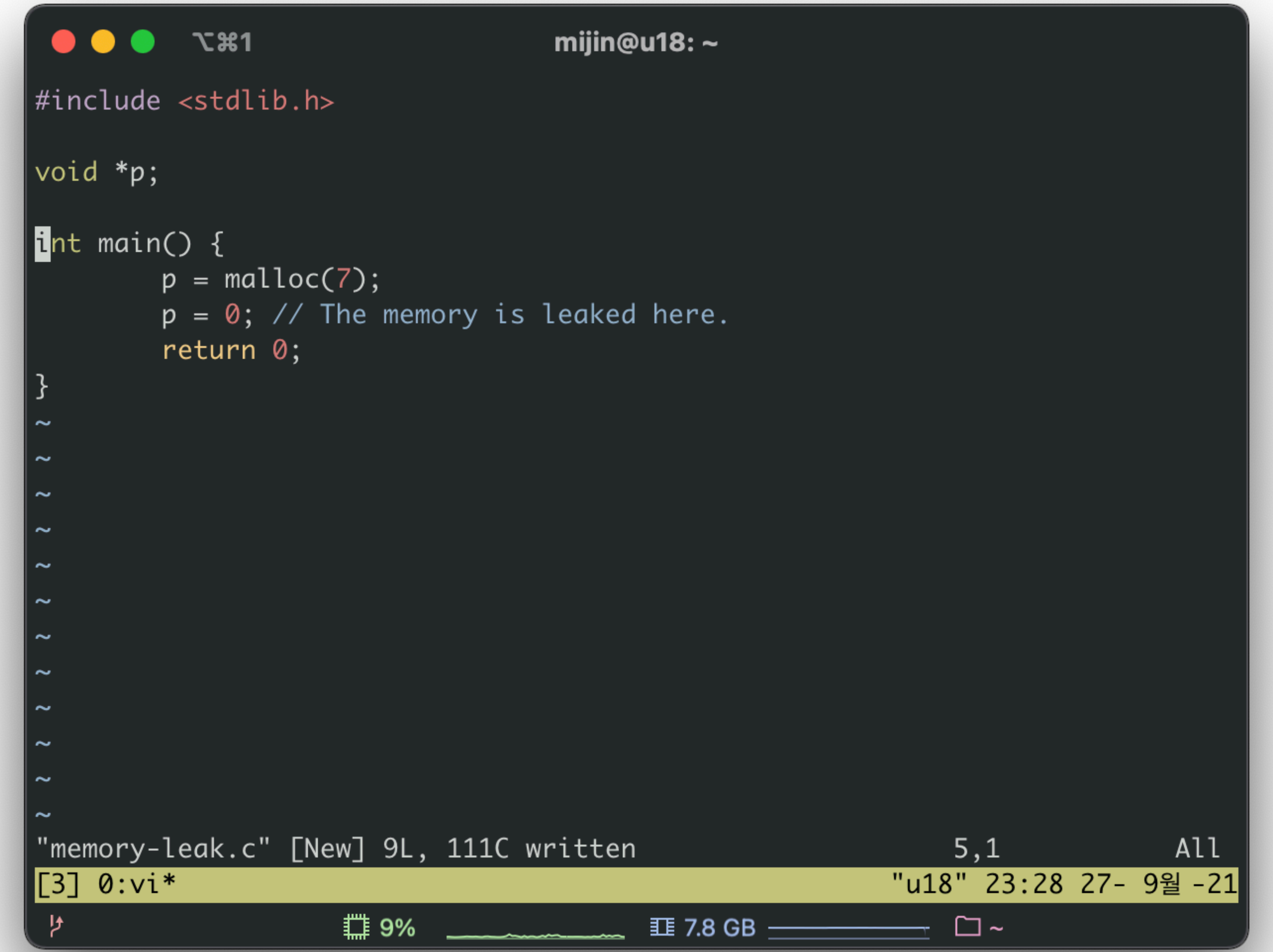


```
#include <iostream>

int main() {
    int *x = new int(10); // no-deleting of a heap-allocated object
    delete x;

    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
"memory-leak.cpp" 8L, 133C written                        6,0-1         All
[3] 0:vi*                                         "u18" 23:35 27- 9월 -21
```

Stand-Alone Mode

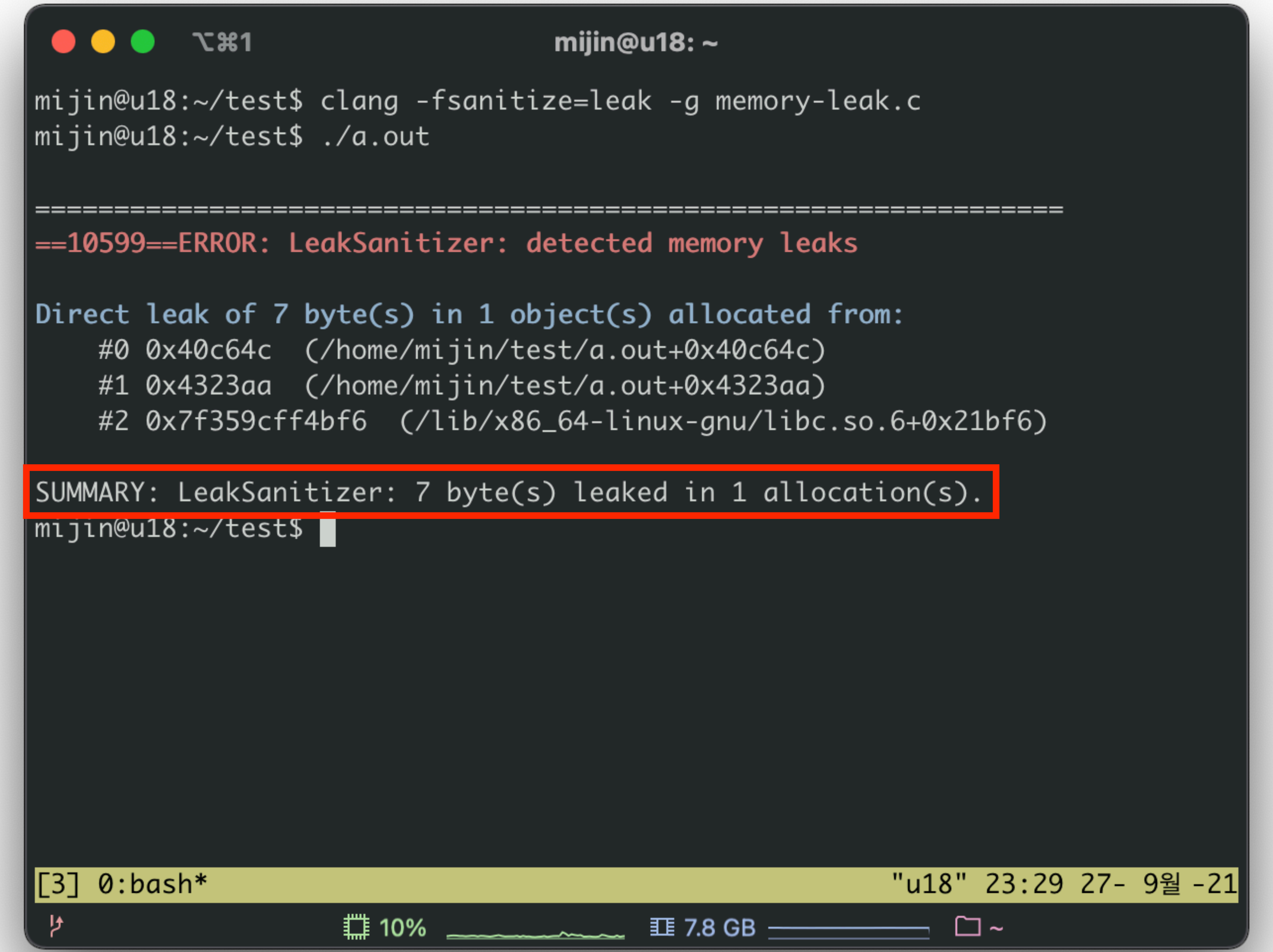
- If you **just need leak detection**
- Or you don’t want to bear the AddressSanitizer **slowdown**
- Then, build the code with `–fsanitize=leak` instead of `–fsanitize=address`

Example on Memory Leaks in C: Code



```
#include <stdlib.h>

void *p;

int main() {
        p = malloc(7);
        p = 0; // The memory is leaked here.
        return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
"memory-leak.c" [New] 9L, 111C written                    5,1           All
[3] 0:vi*                                     "u18" 23:28 27- 9월 -21
```

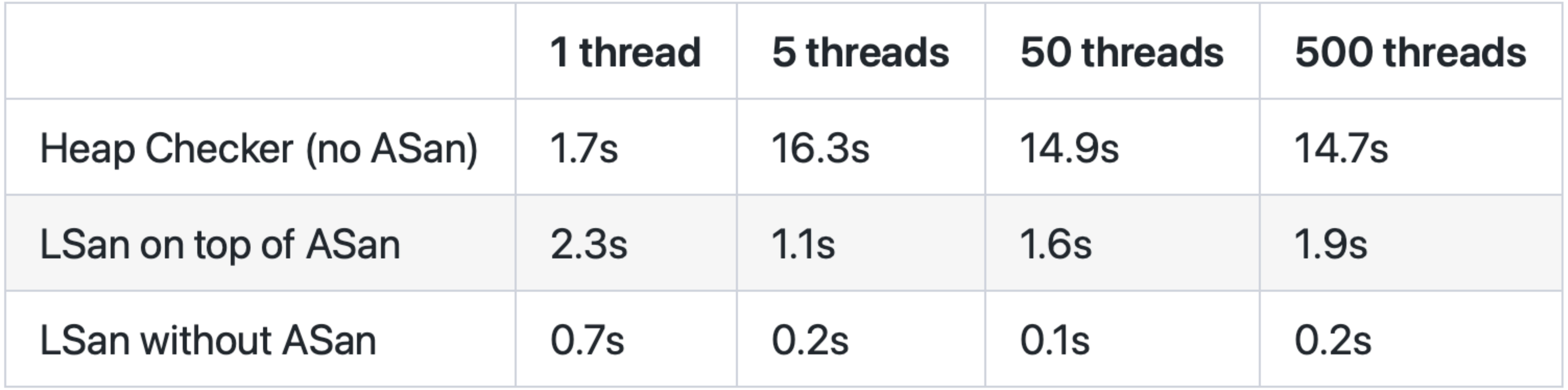
Example on Memory Leaks in C: Result



```
mijin@u18:~/test$ clang -fsanitize=leak -g memory-leak.c
mijin@u18:~/test$ ./a.out

=================================================================
==10599==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 7 byte(s) in 1 object(s) allocated from:
    #0 0x40c64c  (/home/mijin/test/a.out+0x40c64c)
    #1 0x4323aa  (/home/mijin/test/a.out+0x4323aa)
    #2 0x7f359cff4bf6  (/lib/x86_64-linux-gnu/libc.so.6+0x21bf6)

SUMMARY: LeakSanitizer: 7 byte(s) leaked in 1 allocation(s).
mijin@u18:~/test$
```

LeakSanitizer vs. Heap Checker

- LeakSanitizer was designed to replace the gperftools Heap Leak Checker
- LeakSanitizer reports **leaks in the presence of threads correctly** (Heap Checker doesn’t) and supports **more information about leaks**
- A crude synthetic benchmark result:
  - 10 million `malloc/free` pairs in parallel



	1 thread	5 threads	50 threads	500 threads
Heap Checker (no ASan)	1.7s	16.3s	14.9s	14.7s
LSan on top of ASan	2.3s	1.1s	1.6s	1.9s
LSan without ASan	0.7s	0.2s	0.1s	0.2s

Reference

[1] Marc Gregoire, “Professional C++, 4th Edition”, Wiley, 2018

[2] Google, “AddressSanitizerLeakSanitizer”, GitHub (google/sanitizers), https://github.com/google/sanitizers/wiki/AddressSanitizerLeakSanitizer

[3] The Clang Team, “Clang 13 documentation: LEAKSANITIZER”, Clang 13 documentation, https://clang.llvm.org/docs/LeakSanitizer.html

[4] Google, “AddressSanitizerLeakSanitizerVsHeapChecker”, GitHub (google/sanitizers), https://github.com/google/sanitizers/wiki/AddressSanitizerLeakSanitizerVsHeapChecker