

debugging/gcc

Contents

GCC	1
Static-Analyzer	1
Example: Static-Analyzer - Use after free	1
Example: Static-Analyzer - Uninitialized values	3
Sanitizer	3
Example: Sanitizer - Use after free	3
Example: Sanitizer - Division by zero	4
Example: Sanitizer - Datatype Overflow	5
Example: Sanitizer - Data race	6

[[*TOC*]]

GCC

Static-Analyzer

- Static Analyzer Options

Example: Static-Analyzer - Use after free

- Source code use_after_free.c++

```
#include <iostream>

int main(int argc, char *argv[]) {
    char *charArray = new char[10];
    delete [] charArray;
    std::cout << charArray[0] << std::endl;
}
```

- Set up compiler environment
module add compiler/gnu/12
- Compile

```

c++ -fanalyzer use_after_free.c++ -o use_after_free

use_after_free.c++: In function 'int main(int, char**)':
use_after_free.c++:6:29: warning: dereference of NULL 'charArray' [CWE-476] [-Wanalyzer
  6 |         std::cout << charArray[0] << std::endl;
    |                             ~~~~~^
'int main(int, char**)': events 1-5
|
|   4 |         char *charArray = new char[10];
|   |                                     ^
|   |                                     |
|   |                                     (1) allocated here
|   5 |         delete [] charArray;
|   |         ~~~~~
|   |         |
|   |         (2) assuming 'charArray' is NULL
|   |         (3) following 'false' branch (when 'charArray' is NULL)...
|   6 |         std::cout << charArray[0] << std::endl;
|   |         ~~~~~
|   |         |
|   |         (4) ...to here
|   |         (5) dereference of NULL 'charArray'
|
use_after_free.c++:6:29: warning: use after 'delete[]' of 'charArray' [CWE-416] [-Wanalyzer
  6 |         std::cout << charArray[0] << std::endl;
    |                             ~~~~~^
'int main(int, char**)': events 1-6
|
|   4 |         char *charArray = new char[10];
|   |                                     ^
|   |                                     |
|   |                                     (1) allocated here
|   5 |         delete [] charArray;
|   |         ~~~~~
|   |         |
|   |         |
|   |         (4) ...to here
|   |         (5) deleted here
|   |         (2) assuming 'charArray' is non-NULL
|   |         (3) following 'true' branch (when 'charArray' is non-NULL)...
|   6 |         std::cout << charArray[0] << std::endl;
|   |         ~~~~~
|   |         |
|   |         (6) use after 'delete[]' of 'charArray'; delete
|

```

Example: Static-Analyzer - Uninitialized values

- Source code uninitialized_values.cpp

```
#include <iostream>

int main(int argc, char *argv[]) {
    double x, y;
    x = y + 1.;
    std::cout << "x = " << x
               << "y = " << y << std::endl;
}
```

- Set up compiler environment

```
module add compiler/gnu/12
```

- Compile

```
c++ -fanalyzer uninitialized_values.cpp -o uninitialized_values
```

```
uninitialized_values.cpp: In Funktion »int main(int, char**)«:
```

```
uninitialized_values.cpp:5:7: Warnung: use of uninitialized value »y« [CWE-457] [-Wanalyzer-uninit]
```

```
5 |      x = y + 1.;
  |      ~~~~~
»int main(int, char**)«: events 1-3
|
|      4 |      double x, y;
|      |      ^
|      |      |
|      |      (1) region created on stack here
|      |      (2) capacity: 8 bytes
|      5 |      x = y + 1.;
|      |      ~~~~~
|      |      |
|      |      (3) use of uninitialized value »y« here
|
```

Sanitizer

- Wikipedia: Address Sanitizer
- AddressSanitizer with gcc
- Available runtime config options
 - Run-time flags)
 - Environment variable ASAN_OPTIONS="help=1"

Example: Sanitizer - Use after free

- Source code use_after_free.cpp

```
#include <iostream>

int main(int argc, char *argv[]) {
    char *charArray = new char[10];
    delete [] charArray;
    std::cout << charArray[0] << std::endl;
}
```

- Set up compiler environment

```
module add compiler/gnu/12
```

- Compile

```
c++ -fsanitize=address -g use_after_free.cpp -o use_after_free
```

- Execute

```
./use_after_free
```

```
=====
```

```
==131470==ERROR: AddressSanitizer: heap-use-after-free on address 0x602000000010 at pc
READ of size 1 at 0x602000000010 thread T0
```

```
#0 0x55811bc3625c in main .../use_after_free.cpp:6
```

```
#1 0x7ff728c3c78f (/usr/lib/libc.so.6+0x2378f)
```

```
#2 0x7ff728c3c849 in __libc_start_main (/usr/lib/libc.so.6+0x23849)
```

```
#3 0x55811bc36124 in _start (.../use_after_free+0x1124)
```

```
0x602000000010 is located 0 bytes inside of 10-byte region [0x602000000010,0x602000000010)
freed by thread T0 here:
```

```
#0 0x7ff7292c135a in operator delete[](void*) /usr/src/debug/gcc/gcc/libsanitizer/asan/asan.cpp:51
```

```
#1 0x55811bc36228 in main .../use_after_free.cpp:5
```

```
#2 0x7ff728c3c78f (/usr/lib/libc.so.6+0x2378f)
```

```
previously allocated by thread T0 here:
```

```
#0 0x7ff7292c07f2 in operator new[](unsigned long) /usr/src/debug/gcc/gcc/libsanitizer/asan/asan.cpp:51
```

```
#1 0x55811bc36211 in main .../use_after_free.cpp:4
```

```
#2 0x7ff728c3c78f (/usr/lib/libc.so.6+0x2378f)
```

```
...
```

Example: Sanitizer - Division by zero

- Source code division_by_zero.cpp

```
#include <iostream>

int main(int argc, char *argv[]) {
    int a=1, b=0;
    std::cout << a/b << std::endl;
}
```

- Set up compiler environment

```
module add compiler/gnu/12
```
- Compile

```
c++ -fsanitize=undefined -g division_by_zero.cpp -o division_by_zero
```
- Execute

```
./division_by_zero
```

```
division_by_zero.cpp:5:19: runtime error: division by zero
Floating point exception (core dumped)
```
- Execute without sanitizer

```
./division_by_zero
```

```
Floating point exception (core dumped)
```

Example: Sanitizer - Datatype Overflow

- Source code overflow.cpp

```
#include <climits>
#include <iostream>

int main(int argc, char *argv[]) {
    int i = INT_MAX;
    i++;
    std::cout << i << std::endl;
}
```
- Set up compiler environment

```
module add compiler/gnu/12
```
- Compile

```
c++ -fsanitize=undefined -g overflow.cpp -o overflow
```
- Execute

```
./overflow
```

```
overflow.cpp:6:6: runtime error: signed integer overflow: 2147483647 + 1 cannot be represented
-2147483648
```
- Execute without sanitizer

```
./overflow
```

```
-2147483648
```

Example: Sanitizer - Data race

- Source code `data_race.cpp`

```
#include <iostream>

int main () {
    int i = 0;
    #pragma omp parallel shared(i)
    i++;
    std::cout << i << std::endl;
}
```

- Set up compiler environment

```
module add compiler/gnu/12
```

- Compile

```
c++ -fsanitize=thread -fopenmp -g data_race.cpp -o data_race
```

- Execute

```
./data_race
```

```
=====
```

```
WARNING: ThreadSanitizer: data race (pid=132390)
```

```
Write of size 4 at 0x7fff3b830660 by main thread:
```

```
#0 main._omp_fn.0 .../data_race.cpp:6 (data_race+0x1394)
```

```
#1 GOMP_parallel /usr/src/debug/gcc/gcc/libgomp/parallel.c:178 (libgomp.so.1+0x141d0)
```

```
#2 <null> <null> (libc.so.6+0x2378f)
```

```
Previous read of size 4 at 0x7fff3b830660 by thread T2:
```

```
#0 main._omp_fn.0 .../data_race.cpp:6 (data_race+0x137f)
```

```
#1 gomp_thread_start /usr/src/debug/gcc/gcc/libgomp/team.c:129 (libgomp.so.1+0x1d470)
```

```
Location is stack of main thread.
```

```
Location is global '<null>' at 0x000000000000 ([stack]+0x1e660)
```

```
Thread T2 (tid=132393, running) created by main thread at:
```

```
#0 pthread_create /usr/src/debug/gcc/gcc/libsanitizer/tsan/tsan_interceptors_posix.
```

```
#1 gomp_team_start /usr/src/debug/gcc/gcc/libgomp/team.c:858 (libgomp.so.1+0x1dad0)
```

```
#2 <null> <null> (libc.so.6+0x2378f)
```

```
SUMMARY: ThreadSanitizer: data race .../data_race.cpp:6 in main._omp_fn.0
```

```
=====
```

```
1
```

```
ThreadSanitizer: reported 1 warnings
```