C++ Undefined Behavior

What is it, and why should I care?

- A presentation originally by Marshal Clow
- Original: https://www.youtube.com/watch?v=uHCLkb1vKaY
- Original Slides: https://github.com/boostcon/cppnow_presentations_2014/blob/master/files/Undefined-Behavior.pdf
- All errors in this slides or presentation are mine

What is Undefined Behavior?

1.3.24 [defns.undefined]
undefined behavior

behavior for which this International Standard imposes no requirements

[Note: Undefined behavior may be expected when this International Standard omits any explicit definition of behavior or when a program uses an erroneous construct or erroneous data. Permissible undefined behavior ranges from ignoring the situation completely with unpredictable results, to behaving during translation or program execution in a documented manner characteristic of the environment (with or without the issuance of a diagnostic message), to terminating a translation or execution (with the issuance of a diagnostic message).
Many erroneous program constructs do not engender undefined behavior; they are required to be diagnosed.

No requirements

Some examples what can happen

- Program crashes
- Program gives unexpected results
- Computer catches fire
- Daemons fly out of your nose
- Program appears to work fine

- There are no wrong answers!

Example #1
No Wrong Answers!

```
#include <iostream>

 /* what does this program print? */
int main()
{
   int arr[] = { 0, 2, 4, 6, 8 };
   int i = 1;
   std::cout << i + arr[++i] + arr[i++] << “\n”;
}
```

How can I get UB? (1)

- Signed integer overflow (but not unsigned!)
- Dereferencing nullptr (NULL)
- Dereferencing result of malloc(0);
- Shift greater than (or equal to) the width of the operand
- Reading from uninitializer variables
- Modifying a variable more than once in an expression
- Buffer overflow
- Comparing pointers to different data structures

How can I get UB? (2)

- Pointer overflow
- Modifying a const object
- Modifying a string literal
- Negating INT_MIN
- Mismatch between new and delete ([])
- Calling a library routine without fulfilling the prerequisites (z.B. memcpy with overlapping buffers)
- Data races

Example #2

```
#include <new>

class Foo {
    // some complicated class
};

int main()
{
    Foo *p = new Foo[4];
    // lots of stuff here
    delete p;
}
```

atomic_is_lock_free

20.8.2.5 shared_ptr atomic access   [util.smartptr.shared.atomic]
...
template<class T>
bool atomic_is_lock_free(const shared_ptr<T>* p);

Requires: p shall not be null.
...

Arithmetic Operations

5 Expressions [expr]

...

If during the evaluation of an expression, the result is not mathematically defined or not in the range of representable values for its type, the behavior is undefined.

...

Example #3
No Wrong Answers!

```
#include <stdio.h>
#include <stdbool.h>

int main()
{
   bool b;

   if (b)  printf(“true\n”);
   If (!b) printf(“false\n”);
}
```

Why do C and C++ do this?

- It gives the compiler leeway to generate smaller code, by omitting checks
- By assuming no UB the compiler can generate simpler, faster and smaller code
- Different hardware reacts different

Why is this important?

- Because compilers know it – and optimizers take advantage of it
- It is perfectly legal to transform a program exhibiting UB into any other program
- Remember: in UB there are no wrong answers

Different kinds of routines

- Type 1: no UB, no matter what the inputs
- Type 2: UB for some subset of all possible inputs
- Type 3: UB, no matter what the inputs

Example #4

```
int *do_something(
    int *p)
{
    log(“do_something %d”, *p);
    if (!p) {
        …
        p = malloc(...);
        ...
    }
    return p;
}
```

Beispiel #5

```
#include <stdio.h>

int main()
{
   int i = 0x10000000;
   int c = 0;
   do {
      c++;
      i += i;
      printf("%d\n", i);
   } while (I > 0);
   printf("%d iterations\n", c);
}
```

Why do we care? (1)

- It is surprisingly easy to write code with undefined behavior
- http://code.google.com/p/nativeclient/issues/detail?id=245
- UB Code may “work” for a while, and the “break” when optimization level is increased or the compiler is upgraded (“optimization-unstable code” - STACK)

Why do we care? (2)

- UB shows up in “tricky” code; frequently code that is attempting security checks
- http://gcc.gnu.org/bugzilla/show_bug.cgi?id=30475
- Bugs that STACK found in Postgres

Example #6

```
// Checks are hard to write correctly
// From “Apple Secure Coding Guidelines” (second edition)

void xxxx(
    int n,
    int m)
{
    size_t bytes = n * m;
    if (n > 0 && m > 0 && SIZE_MAX / n >= m) {
        …
        /* allocate “bytes” space */
        …
    }
}
```

Beispiel #7
Aliasing

```
struct Foo {
   int a;
};
struct Bar {
   int a;
   int b;
};

Foo f{3};
Bar *p = (Bar *)&f;
p->a = 4;
std::cout << f.a << “\n”;
```

What can I do about UB?

- Be aware of UB
- Don’t blame the compiler (“don't shoot the messenger”)
- When you do something tricky think about UB
- Build your code with several compilers and optimization levels

Impact

If you use Undefined Behavior you con no longer reason about what your program does

It is too late

You can not check if Undefined Behavior has already happened

Example #8

```
// wrong
bool will_this_overflow(int a)
{
    return a + 100 < a;
}
// what the compiler can/will generate:
bool will_this_overflow(int a)
{
    return false;
}
```

```
// correct
bool will_this_overflow(int a)
{
    return a < (INT_MAX - 100);
}
```

Example #9

```
#include <sstream>


void xxxx(
   int fh,
   int i)
{
   std::stringstream ss;
   ss << I << “\n”;
   write(fh, ss.str().c_str(), ss.str().size());
}
```

Tools

- Clang: -fsanitize=undefined
- John Regehr's Integer Overflow Checker
- STACK

Quiz

```
// Optimize this code

void contains_null_check(
    int *p)
{
    int dead = *p;
    if (p == nullptr)
        return;
    *p = 4;
}
```

References

- A Guide to Undefined Behavior in C and C++, Part 1
  http://blog.regehr.org/archives/213 (links zu Teil 2 und 3)
- Towards optimization-safe systems http://pdos.csail.mit.edu/papers/stack:sosp13.pdf
- What every C programmer should know about undefined behavior
  http://blog.llvm.org/2011/05/what-every-c-programmer-should-know.html
- It's time to get serious about exploiting Undefined Behavior
  http://blog.regehr.org/archives/761
- Finding Undefined Behavior by finding dead code http://blog.regehr.org/archives/970
- About unspecified and undefined behavior in C (ACCU 2013)
  http://www.pvv.org/~oma/UnspecifiedAndUndefined_ACCU_Apr2013.pdf