

Developing Correct C++ @ Scale

Mark Isaacson

Fundamental Developer Concerns

- How do you prevent code from breaking?
- How do you define “broken”?
- How do you know if something broke?
- How do you figure out why something broke?
- Who do you talk to when something breaks?

Perspective

Facebook

- Thousands of developers
- 1 repo = most C++ projects
- Hundreds of thousands of files
- Broken code can lead to a broken product
- Continued development/release

Setting the Scene

```
void enforceIsFive(int x) {  
    EXPECT_EQ(x, 5);  
}
```

```
TEST(MySillyTest, BasicTest) {  
    enforceIsFive(5);  
    enforceIsFive(3); // Oops  
}
```

Setting the Scene

```
[=====] Running 1 test from 1 test case.
```

```
SillyTest.cpp:31: void enforceIsFive(int): Expect `x == 5' failed.
```

Setting the Scene

```
[=====] Running 1 test from 1 test case.
```

```
SillyTest.cpp:31: void enforceIsFive(int): Expect `x == 5' failed.
```

```
*** Signal 6 (SIGABRT) stack trace: ***
```

```
@ enforceIsFive(int)
```

```
@ SillyTest_BasicTest_Test::TestBody()
```

```
@ RUN_ALL_TESTS()
```

A default test main function

```
int main(int argc, char** argv) __attribute__((__weak__));
```

```
int main(int argc, char** argv) {  
    ::testing::InitGoogleTest(&argc, argv);  
    folly::symbolizer::installFatalSignalHandler();  
    /* Stuff */  
    folly::symbolizer::installFatalSignalCallbacks();  
    return RUN_ALL_TESTS();  
}
```

Code Review

How do you know that nothing broke?

Automatic Code Review

```
279     if (target.size() >= 4) {
280         base64_encode_source_q_pull(ctx, source, target);
281     } else if (target.size() > 0) {
282         // encode to stack buffer
283         const auto n = target.size();
```

ClangWarning-Wunused-variable

 Report



In file included from folly/Base64.cpp:1:

./folly/Base64.h:283:16: warning: unused variable 'n'

```
const auto n = target.size();
           ^
```

```
284     auto mid = std::array<uint8_t, 4>{};
285     auto midr = range(mid);
```

Automatic Code Review

Builds & Tests

diff sanity

- BUILD FAILURE** 1
 - proxygen
- TEST FAILURE** 2
 - tutorials
 - wangle
- NO TEST SIGNAL** 1
 - thrift
- SUCCESS** 3
 - folly
 - folly_ubsan
 - zstd

Finished in **8 minutes**

folly finished without errors, but possible problems were detected with other jobs. Please investigate any unsuccessful jobs before proceeding.

For full details for **folly**, see [full job](#)

Pass/Fail State is complicated

- Green is “nothing new”

 **SUCCESS**

Pass/Fail State is complicated

- Green is “nothing new”
- Wardens keep signal high

● SUCCESS

TEST FAILURE DETAILS

Graph:

Test History; drag to zoom



Automatic Code Review

Builds & Tests

diff **sanity**

- BUILD FAILURE** 1
 - proxygen
- TEST FAILURE** 2
 - tutorials
 - wangle
- NO TEST SIGNAL** 1
 - thrift
- SUCCESS** 3
 - folly
 - folly_ubsan
 - zstd

Finished in **8 minutes**





folly finished without errors, but possible problems were detected with other jobs. Please investigate any unsuccessful jobs before proceeding.

For full details for **folly**, see [full job](#)

Automatic Code Review

Builds & Tests

diff sanity

- **BUILD FAILURE** 1
 - folly 
- **TEST FAILURE** 2
 - tutorials 
 - wangle 
- **NO TEST SIGNAL** 1
 - thrift 

folly/test/ForeachTest.cpp:23:54 Failure
Unknown type name 'in'; did you mean 'int'?

For full details, see [full logs](#).

There are **5 users** affected by this error today.

Human Code Review

- Human component
 - Interpreting the automatic information
 - Meta-correctness checks
 - The ultimate decider

Shipping the Code



Ship It

Shipping the Code

Land Confirmation ×

Whoa! You probably don't want to land this diff yet:

- 1) This diff is more than 5 days old. You should **really** rebase this and wait for builds before landing it.
- 2) Tests Failed

Are you sure you want to Land:

Optimizing Turnaround

Builds & Tests

sanity

● BUILD FAILURE 1

sanity check



What gets through?

- Undefined behavior
- Time/load dependent failures

ASAN

```
int main() {  
    auto x = new int(5);  
    delete x;  
    *x = 3;  
    return 0;  
}
```

ASAN

ERROR: AddressSanitizer: heap-use-after-free

WRITE of size 4 at 0xf0 thread T0

```
#0 my_binary.cpp:9 main
```

0xf0 is located 0 bytes inside of 4-byte region [0xf0,0xf4)

freed by thread T0 here:

```
#0 my_binary+0x70 operator delete(void*)
```

```
#1 my_binary.cpp:8 main
```

previously allocated by thread T0 here:

```
#0 my_binary+0x30 operator new(unsigned long)
```

```
#1 my_binary.cpp:7 main
```

ASAN (Raw-ish Output)

```
ERROR: AddressSanitizer: heap-use-after-free on address 0xf0
at pc 0x19 bp 0xa0 sp 0x98
```

```
WRITE of size 4 at 0xf0 thread T0
```

```
#0 0x18 in main my_binary.cpp:9
```

```
#1 0xf5 in __libc_start_main (lib/libc.so.6+0xf5)
```

```
#2 0x25 in _start sysdeps/x86_64/start.S:12
```

```
0xf0 is located 0 bytes inside of 4-byte region [0xf0,0xf4)
```

ASAN (Raw-ish Output)

freed by thread T0 here:

```
#0 0x70 in operator delete(void*) (my_binary+0x40)
#1 0xd5 in main my_binary.cpp:8
#2 0xf5 in __libc_start_main (lib/libc.so.6+0xf5)
#3 0x25 in _start sysdeps/x86_64/start.S:12
```

previously allocated by thread T0 here:

```
#0 0x30 in operator new(unsigned long) (my_binary+30)
#1 0x6a in main my_binary.cpp:7
#2 0xf5 in __libc_start_main (lib/libc.so.6+0xf5)
#3 0x25 in _start sysdeps/x86_64/start.S:12
```

ASAN (Reduced Output)

ERROR: AddressSanitizer: heap-use-after-free

WRITE of size 4 at 0xf0 thread T0

```
#0 my_binary.cpp:9 main
```

0xf0 is located 0 bytes inside of 4-byte region [0xf0,0xf4)

freed by thread T0 here:

```
#0 my_binary+0x70 operator delete(void*)
```

```
#1 my_binary.cpp:8 main
```

previously allocated by thread T0 here:

```
#0 my_binary+0x30 operator new(unsigned long)
```

```
#1 my_binary.cpp:7 main
```


miner

Your job has `user error` .

Here is my best guess of what is wrong:

```
LocatorTest.cpp:929: Failure
```

```
Value of: cpSingle.isFallback()
```

```
Actual: false
```

```
Expected: true
```

```
LocatorTest.cpp:936: Failure
```

```
Value of: dbname
```

```
Actual: "IPPORT|0000|9865|test"
```

```
Expected: cpSingle.getDbname()
```

```
Which is: "test"
```

* This code/test output is fictitious.

Mining Error Messages

Informing Developers

- Repro command
- Error message
- Test configuration
- History
- # of breakages

1 problem, 1 task

Task Aggregation

- By configuration
- By binary
- By "fail rev"

Who knows about X?

Credits

- Developer Infrastructure
 - Product Stability
 - Signal Infrastructure
 - fbcode Foundation
- Security Infrastructure
- + More

Fundamental Developer Concerns

- How do you prevent code from breaking?
- How do you define “broken”?
- How do you know if something broke?
- How do you figure out why something broke?
- Who do you talk to when something breaks?

```
return 0;
```