

The slide features a light gray background with two prominent diagonal stripes. One stripe is dark gray and runs from the top-left corner towards the bottom-right. The other stripe is bright yellow and runs from the bottom-left corner towards the top-right. These stripes intersect to create a white triangular area in the center-left of the slide.

chaincode

Debugging Bitcoin

Fabian Jahr

DISCLAIMER

THIS DRAMA IS INSPIRED BY TRUE EVENTS BUT IT IS NOT A DOCUMENTARY/BIOGRAPHY OF ANY CHARACTER DEPICTED IN THE DRAMA. NO SCENES SHOULD BE CONSTRUED TO REPRESENT A TRUE OR ACCURATE RECREATION OF THE ACTUAL EVENTS THAT TRANSPIRED. THE STORY AND THE RELATIONSHIPS BETWEEN THE CHARACTERS DEPICTED IN THE DRAMA HAVE BEEN FICTIONALIZED. ANY INSINUATION OR RESEMBLANCE OF ANY CHARACTER TO ANY PERSON, DEAD OR ALIVE, OR THE RESEMBLANCE OF THE DRAMA STORY TO ANY REAL STORY IS PURELY COINCIDENTAL.

**CIGARETTE SMOKING, ALCOHOL AND TOBACCO CONSUMPTION
ARE INJURIOUS TO HEALTH**

Welcome to Bitcoin

jobs

- ☒ sales ¹¹
- ☒ accounting/finance ⁴
- ☒ education/teaching ³
- ☒ customer service ²
- ☒ writing/editing ²
- + show 26 more ⁷
- ☒ select all

- ☐ search titles only
- ☐ has image
- ☐ posted today
- ☐ bundle duplicates
- ☐ include nearby areas

MILES FROM ZIP

miles from zip ↻



remote



- ★ Jul 22 [Hiring Sales Superstars \\$100K plus LIFETIME RESIDUAL](#) (New York) [pic](#) [img](#)
- ★ Jul 20 [WORK FROM HOME: ADMINS WANTED](#) (REMOTE)
- ★ Jul 20 [Part-time Bookkeeping position \(Manhattan & Staten Island\)](#)
- ★ Jul 19 [Kick-Ass Sales Rep for Fast-Growing NYC Startup](#) (New York, NY) [pic](#)
- ★ Jul 18 [Junior Bitcoin Core Dev](#) (remote)
- ★ Jul 17 [Customer Service and Event Support](#) (Telework)
- ★ Jul 9 [Finance Tutor \(PT\) | \\$12 - \\$16 per hour](#) (remote)

First issues

```
3  
2   LogPrintf("@@@@@@@@@@@@");  
1   std::cout << "#####<img alt="A yellow and grey diagonal decorative bar in the top right corner of the slide." data-bbox="720 0 1000 480"/>" <<std::endl;
```

```
$ cat ~/Library/Application\ Support/Bitcoin/debug.log | grep @@@
```

wat?

Let's ask in the chat

1

1337hax0r

Hey, the logging doesn't work!!!

Helpful senior dev is helpful

```
$ bitcoin-cli -regtest getdifficulty  
4.656542373906925e-10
```

std::out

```
2019-08-05T17:36:14Z Received a POST request for / from 127.0.0.1:65457  
2019-08-05T17:36:14Z ThreadRPCServer method=getdifficulty user=satoshi  
2019-08-05T17:36:14Z @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
tor: Error connecting to Tor control socket
```

regtest/debug.log

```
$ sudo cat ~/Library/Application\ Support/Bitcoin/regtest/debug.log | grep @@@  
2019-08-05T17:27:19Z @@@@@@@@@@@@@@@@@@Adding fixed seed nodes as DNS doesn't seem to be available.  
2019-08-05T17:34:47Z @@@@@@@@@@@@@@@@@@tor: Error connecting to Tor control socket  
2019-08-05T17:36:14Z @@@@@@@@@@@@@@@@@@tor: Error connecting to Tor control socket
```

Moving on to unit tests



MarcoHawk commented 5 hours ago

Maybe add some unit tests?

To print messages in the unit tests, you cannot use `LogPrintf`. Instead use `BOOST_TEST_MESSAGE` and `BOOST_CHECK_MESSAGE` to print messages from the tests. You will have to run the test binary directly (`src/test/test_bitcoin`) with `--log_level=all` to see the messages.

To print from the source files themselves, you could use `fprintf()` and print your messages to `stderr`.

[share](#) [edit](#) [flag](#)

edited Jun 25 '18 at 20:40

answered Jun 25 '18 at 7:36



Andrew Ciao ♦

37.4k ● 4 ● 28 ● 68

That was kind of easy

```
delete block_index;  
BOOST_TEST_MESSAGE("FOO");  
  
RejectDifficultyMismatch(difficulty, expect
```

```
$ src/test/test_bitcoin --run_test=blockchain_tests --log_level=all  
Running 5 test cases...  
Entering test module "Bitcoin Core Test Suite"  
test/blockchain_tests.cpp:46: Entering test suite "blockchain_tests"  
test/blockchain_tests.cpp:48: Entering test case "get_difficulty_for  
FOO  
test/blockchain_tests.cpp:30: info: check 'Difficulty was 0.000001 h
```


What about using a debugger?

`gdb` or `lldb` on macOS

Start debugger with an executable

Set breakpoints

Run the executable from the debugger

Inspect variables, step through lines etc.

This is pretty cool!

```
$ lladb src/test/test_bitcoin
(lladb) target create "src/test/test_bitcoin"
Current executable set to 'src/test/test_bitcoin' (x86_64).
(lladb) b test/blockchain_tests.cpp:48
Breakpoint 1: 5 locations.
(lladb) run --run_test=blockchain_tests
Process 46577 launched: '/Users/FJ/projects/cpp/bitcoin/src/test/test_bitcoin'
test_bitcoin was compiled with optimization - stepping may behave oddly; stepping one line at a time may be more useful.
Process 46577 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1 hit
    frame #0: 0x0000000100093d42 test_bitcoin`_GLOBAL__sub_I_blockchain_tests.cpp.o
    45
    46 BOOST_FIXTURE_TEST_SUITE(blockchain_tests, BasicTestingSetup)
    47
-> 48 BOOST_AUTO_TEST_CASE(get_difficulty_for_very_low_target)
    49 {
    50     TestDifficulty(0x1f111111, 0.000001);
    51 }
Target 0: (test_bitcoin) stopped.
(lladb)
```

Should not be too hard for functional tests...

Using Python

Logging

- `self.log.debug()`

Debugging

- `import pdb; pdb.set_trace()`

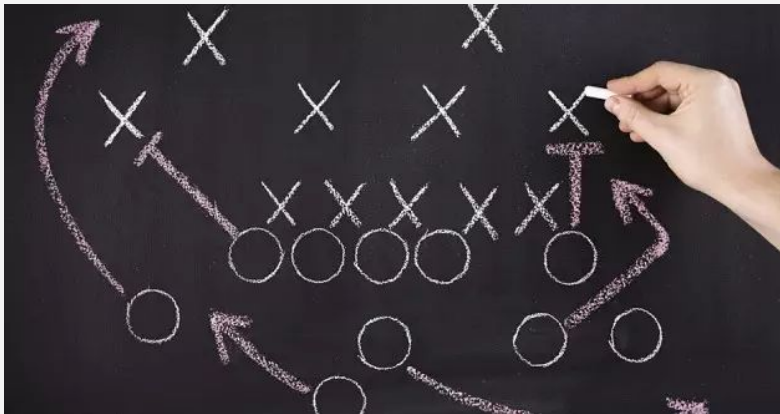
But what about debugging the C++ code?

But where is the executable?

Functional tests launch our `bitcoind` themselves using a temp folder as datadir

That means we can not simply start it ourselves

We need a gameplan!



Gameplan

1. Start the functional test directly (not using `test_runner.py`) and let them start the `bitcoind` process
2. Pause the functional tests with `pdb.set_trace()`
3. Find the running `bitcoind` process, attach to it using `lldb` and setting breakpoints
4. Then let the test continue (`continue` in `pdb`) and let it run into our `lldb` breakpoints
5. Optional: May want to remove 60s timeout



```

9 class BlockchainTest(BitcoinTestFramework):
8     def set_test_params(self):
7         self.setup_clean_chain = True
6         self.num_nodes = 1
5
4     def run_test(self):
3         self.mine_chain()
2         self.restart_node(0, extra_args=['-stop
1
57 import pdb; pdb.set_trace()
1 self._test_getblockchaininfo()
2 self.test_getchainxstats()

```

```

$ ./test/functional/rpc_blockchain.py --loglevel=debug
2019-08-05T19:49:54.334000Z TestFramework (DEBUG): PRNG seed is: 7714090491127031
2019-08-05T19:49:54.334000Z TestFramework (DEBUG): Setting up network thread
2019-08-05T19:49:54.335000Z TestFramework (INFO): Initializing test directory /var
2019-08-05T19:49:54.340000Z TestFramework.node0 (DEBUG): bitcoind started, waitin
2019-08-05T19:49:54.859000Z TestFramework.node0 (DEBUG): RPC successfully started
2019-08-05T19:49:54.925000Z TestFramework (INFO): Create some old blocks
2019-08-05T19:49:55.458000Z TestFramework.node0 (DEBUG): Stopping node
2019-08-05T19:49:55.783000Z TestFramework.node0 (DEBUG): Node stopped
2019-08-05T19:49:55.790000Z TestFramework.node0 (DEBUG): bitcoind started, waitin
2019-08-05T19:49:56.303000Z TestFramework.node0 (DEBUG): RPC successfully started
> /Users/FJ/projects/cpp/bitcoin/test/functional/rpc_blockchain.py(58)run_test()
-> self._test_getblockchaininfo()
(Pdb) continue
2019-08-05T19:50:23.047000Z TestFramework (INFO): Test getblockchaininfo

```

```

$ PATH=/usr/bin:/usr/bin/lldb -p $(pgrep bitcoind)
(lldb) process attach --pid 47683
Process 47683 stopped
* thread #1, name = 'bitcoin-init', queue = 'com.apple.main-thread'
  frame #0: 0x00007fff5cee186a libsystem_kernel.dylib`__psynch_cvwait:
libsystem_kernel.dylib`__psynch_cvwait:
-> 0x7fff5cee186a <+10>: jae     0x7fff5cee1874      ; <+20>
    0x7fff5cee186c <+12>: movq   %rax, %rdi
    0x7fff5cee186f <+15>: jmp     0x7fff5cede457      ; cerro
    0x7fff5cee1874 <+20>: retq
Target 0: (bitcoind) stopped.

Executable module set to "/Users/FJ/projects/cpp/bitcoin/src/bitcoi
Architecture set to: x86_64h-apple-macosx-.
(lldb) b getblockchaininfo
Breakpoint 1: where = bitcoind`getblockchaininfo(JSONRPCRequest co
memory:2230, address = 0x000000010d8e2c9c
(lldb) continue
Process 47683 resuming
bitcoind was compiled with optimization - stepping may behave oddl
Process 47683 stopped
* thread #9, name = 'bitcoin-httpworker.2', stop reason = breakpoint
  frame #0: 0x000000010d8e2c9c bitcoind`getblockchaininfo(JSONRPC
at memory:2139:66 [opt]
2136     typedef const _Tp& const_reference;
2137
2138 #ifndef _LIBCPP_CXX03_LANG
-> 2139     _LIBCPP_INLINE_VISIBILITY constexpr __compressed_pair_el
2140
2141     template <class _Up, class = typename enable_if<
2142         !is_same<__compressed_pair_elem, typename decay<_Up>
Target 0: (bitcoind) stopped.

```

Major to success: Context awareness



Debugging contexts

	Test driver	Bitcoind context
Manual	<ul style="list-style-type: none">- <code>bitcoin-cli/rpc</code>	<ul style="list-style-type: none">- Path: your own bitcoin path- Log: <code>ENV/debug.log</code>- Debug: run <code>bitcoind</code> with <code>lldb</code>
Unit tests	<ul style="list-style-type: none">- <code>src/test/test_bitcoin</code>	<ul style="list-style-type: none">- Path: <code>/var/</code>- Log: to <code>std::out</code> with LibBoost- Debug: Run <code>test_bitcoin</code> with LLDB
Functional tests	<ul style="list-style-type: none">- <code>test/functional/test_runner.py</code> (or the test directly)- Log: <code>self.log.print()</code>- Debug: <code>pdb</code>	<ul style="list-style-type: none">- Path: <code>/var/</code> with <code>--no-cleanup</code>- Log: temporary <code>debug.log</code> with consolidation tool- Debug: <code>pdb + lldb</code>

Things left out

Install `ccache`

Compiler flags

- Disable optimizations (`-O0`)
- Remove parts you don't need, e.g. `--without-gui` for example

Segfault tools

- Core dumps
 - Need to activate with `ulimit -c unlimited` and then run in same terminal session
 - Find them in `/cores/`
 - Make sure to clean up afterwards
- `valgrind`
 - Inspections, used similar to `lldb`



<http://bit.ly/debugbitcoin>

=> work in progress

=> help me with linux version!

Thank you and questions?