Debugging with LLDB

Sciware — March 4, 2021

Nils Wentzell

SIMONS FOUNDATION

FLATIRON INSTITUTE
Center for Computational Quantum Physics

Avoiding bugs

- Expressive Code (Modern C++)
- Code Review
- Automated Tests (googletest, TDD)
- Version Control (git)
- Static analyzer's (clang-tidy)
- Compiler Warnings
- Use dynamic analyzer tools to catch them as they appear!
  - Valgrind
  - LLVM Sanitizers

LLDB — The LLVM Debugger

- Open-Source (Apache 2)
- Easy Setup on both Linux & MacOS
- Language Support: C, C++, Objective-C, Swift
- High Performance
- Great printers for builtins and C++ STL
- Easily scriptable using Python
- Seamless integration with LLVM Sanitizers
- Good Multi-Threading support

Load it on Rusty

```
$ module load llvm/11.0.0
```

LLDB — Starting LLDB

```
$ lldb my_prog

$ lldb -- my_prog args…

$ lldb -p pid
```

LLDB — Printing

- `print var/expr` (`p var/expr`)
- `frame variable` (`fr v`)

LLDB — Navigation

- `run` (r)
- `step` (s)
- `next` (n)
- `continue` (c)
- `thread step-out` (finish)
- `thread backtrace` (bt)
- `frame select` (f)

LLDB —breakpoint (br)

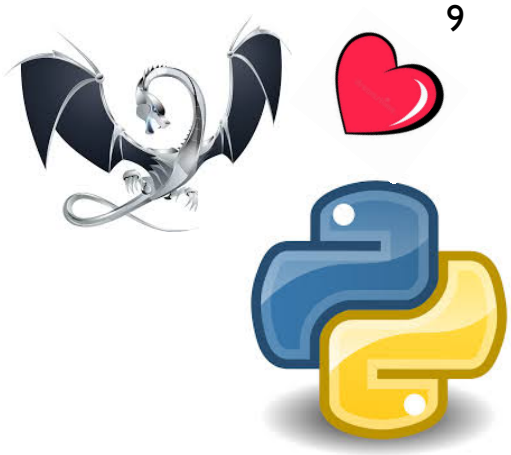
- b src_file:line
- b function_name
- br list
- br delete *br_id*
- br enable *br_id*
- br disable *br_id*
- br modify -c "*condition*"
- br command add *br_id*
- br command list *br_id*

LLDB — watchpoint (wa)

- wa set var *var_name*
- wa list
- wa delete *wa_id*
- wa enable *wa_id*
- wa disable *wa_id*
- wa modify -c “condition” *wa_id*
- wa command add *wa_id*
- wa command list *wa_id*

Let’s see it in action!

LLDB — Script



lldb.llvm.org/use/python

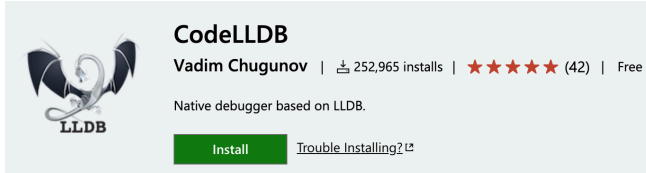
lldb.llvm.org/use/variable

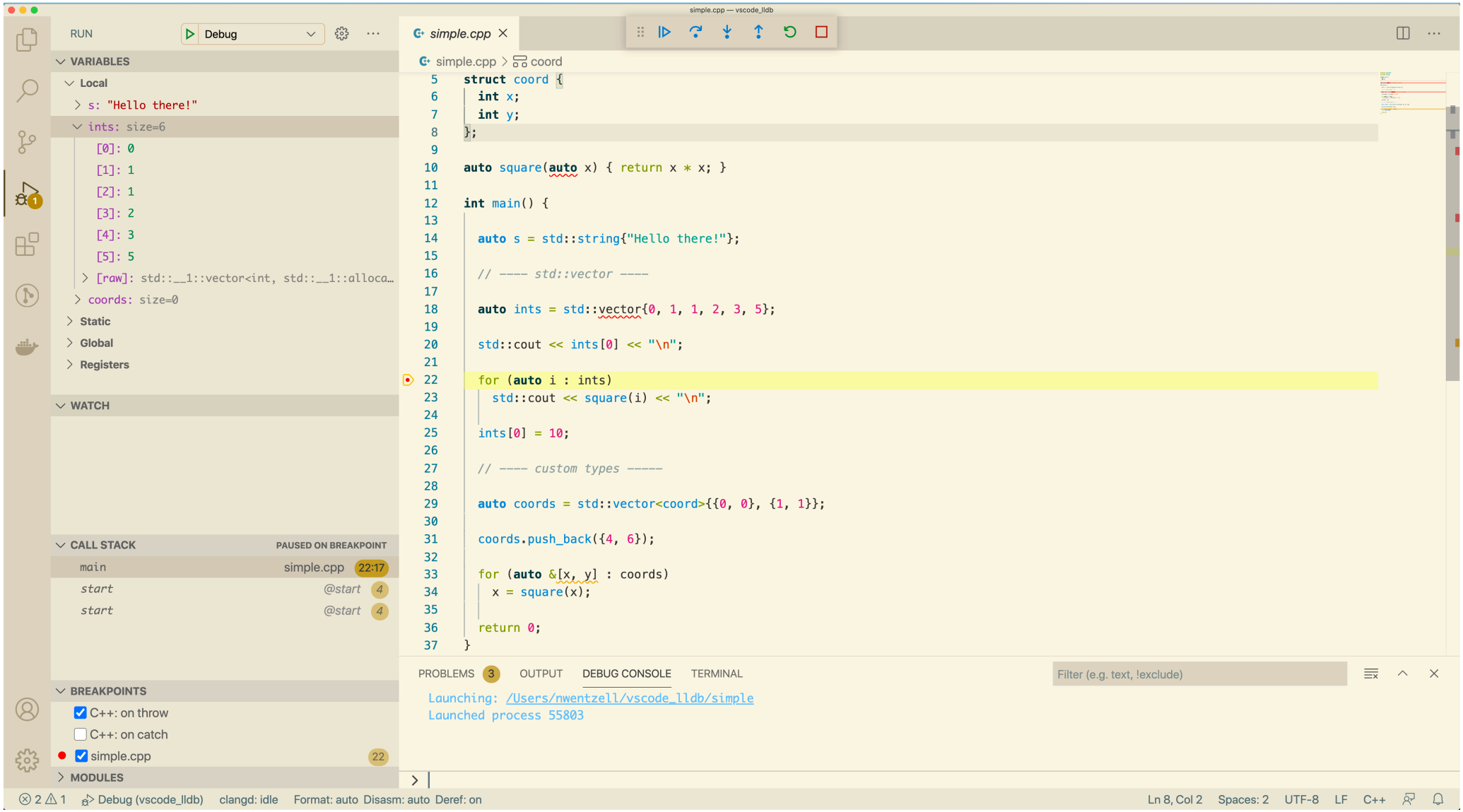
- `script` — Start Python Interpreter
- `type summary add --summary-string “${var.x}“ MyType`
- `br command add -s python` *`br_id`*

Add custom formatters to your

`~/.lldbinit`

LLDB — VSCode


CodeLLDB
Vadim Chugunov | 252,965 installs | (42) | Free
Native debugger based on LLDB.
Install
Trouble Installing?
LLDB


simple.cpp — vscode_lldb
RUN
Debug
VARIABLES
Local
s: "Hello there!"
ints: size=6
[0]: 0
[1]: 1
[2]: 1
[3]: 2
[4]: 3
[5]: 5
[raw]: std::__1::vector<int, std::__1::alloca…
coords: size=0
Static
Global
Registers
WATCH
CALL STACK
PAUSED ON BREAKPOINT
main simple.cpp 22:17
start @start 4
start @start 4
BREAKPOINTS
C++: on throw
C++: on catch
simple.cpp 22
MODULES
simple.cpp
simple.cpp > coord
struct coord {
int x;
int y;
};
auto square(auto x) { return x * x; }
int main() {
auto s = std::string{"Hello there!"};
// ---- std::vector ----
auto ints = std::vector{0, 1, 1, 2, 3, 5};
std::cout << ints[0] << "\n";
for (auto i : ints)
std::cout << square(i) << "\n";
ints[0] = 10;
// ---- custom types -----
auto coords = std::vector<coord>{{0, 0}, {1, 1}};
coords.push_back({4, 6});
for (auto &[x, y] : coords)
x = square(x);
return 0;
}
PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL
Filter (e.g. text, !exclude)
Launching: /Users/nwentzell/vscode_lldb/simple
Launched process 55803
Debug (vscode_lldb)
clangd: idle
Format: auto Disasm: auto Deref: on
Ln 8, Col 2
Spaces: 2
UTF-8
LF
C++

LLDB — Resources

- Basic Tutorial — lldb.llvm.org/use/tutorial
- GDB Command Map — lldb.llvm.org/use/map
- Cheat Sheet — bit.ly/2PpPnd8
- CodeLLDB VSCode Extension — github.com/vadimcn/vscode-lldb

Thank you for your attention!