Dynamic C++

https://github.com/JonChesterfield/dynamic_cpp_talk.git

October 26, 2016

Who are you listening to?

- Software engineer
  - C++ by income
  - C by time spent
  - Lisp by choice
- Mostly writes
  - Toolchains
  - Datastructures
  - Tests

GRAPHCORE

accelerated intelligence

Dynamic?

- Dynamic typing
- Introspection
- Just in time
- Mutable metaclasses
- Object orientation
- eval-in-environment
- read-eval-print-loop

Static?

- Static typing
- Template metaprogramming
- Compilation
- Immutable classes
- Class orientation?
- Embedded data
- Overnight builds

My background languages

- Assembly
- C
- C++
- Fortran
- Javascript
- Lisp
- Matlab
- Python

Ease of use?

Because I want to get stuff done

1. Javascript
2. Matlab
3. Python
4. Fortran
5. C
6. C++
7. Lisp
8. Assembly

Flexibility?

Because I want to do difficult stuff

1. Assembly
2. Lisp
3. Javascript
4. Python
5. C++
6. C
7. Matlab
8. Fortran

Correlation?

1. Javascript
2. Matlab
3. Python
4. Fortran
5. C
6. C++
7. Lisp
8. Assembly

1. Assembly
2. Lisp
3. Javascript
4. Python
5. C++
6. C
7. Matlab
8. Fortran

Expected performance?

In my experience. YMMV!

1. Fortran
2. C++
3. C
4. Matlab
5. Python
6. Javascript
7. Lisp
8. Assembly

Great parts of C++

Great parts of C++

- STL
- atomics
- auto
- catch
- exceptions
- llvm
- malloc
- performance
- preprocessor
- raii
- templates
- valgrind

Terrible parts of C++

Terrible parts of C++

- aliasing
- classes
- compilation
- complexity
- inconsistency
- introspection
- macros
- parsing
- redundancy
- scoping
- the abi
- types

Easy fixes

- -fno-strict-aliasing

C++ static typing example

Persistent hashed trie.

- Represent varint pascal string
- Wanted a uint6_t
- Typesafe conversions to uint8_t
- Implicit type conversions
- Cannot represent contracts

Type systems in C++

- Types on variables, not values
- Structural typing in templates
- Erasure(std::function, void *)
- Opaque types(char[], foo *)
- Nominal polymorphism dispatch
- Overloaded return type via proxy
- Non-transitive const & mutable

Macros | code generators

- ln src/x.c gen/x.c
- python src/x.c.py > gen/x.c
- clang -c gen/x.c -o obj/x.bc

src	gen	obj
x.hpp	x.hpp	
x.cpp	x.cpp	x.bc
x.hpp.py	x.hpp	
x.cpp.py	x.cpp	x.bc

How does LLVM help?

- Clang gives C++ => bitcode
- lli is an interpreter for bitcode
- llvm-link is $x.bc + y.bc => z.bc$
- opt runs optimisation passes
  - The built in ones
  - Any you decide to write
- llc gives bitcode => $x86\_64$

Tooling

- clang
- rtags
- clang-format
- mcjit, orc jit
- domain specific optimisation
- various code sanitizers
- custom compilers

Compilation

- Your library/program is a dir tree
- clang turns leaf .cpp into bc
- llvm-link turns directories into bc
- opt reduces the size of bc
    - internalise symbols here
- recurse...
- base case is one bc into library

Language bindings to C++

- Parse C++ yourself? Use swig?
- Write the class api in .json?
  - Generate the C++ header
  - Generate the language bindings
- Parse the bitcode instead
  - Much friendlier than C++
  - Parser already exists
  - Maps directly to machine types

C++ with Python

- C++ can embed an interpreter
- Python can load a dyn library
- pybind11 makes the latter nicer
- 'iterators' similar enough
- refcounting + raii cooperate
- interactive testing via Python?
- writing hot loops in C++?

The GIMP

- Core implemented in C
- Plugins can be written in
  - C | C++
  - Scheme
  - Python
  - Perl
- Core provides a set of types
- Each plugin provides functions
- Any plugin can call any function

Emacs, circa 1980

- Core implemented in (nasty) C
- Mostly written in lisp
- Dynamically typed
- Dynamically scoped!
- Extensible at runtime
- Ridiculously large codebase
- (Somehow) still runs correctly

Your C++ IDE should...

- Jump to definition
- Show all call sites
- Highlight syntactic errors
- Compile & test on single key
- Automate source formatting
- Interop with source control
- Provide macros

Dynamic C++

https://github.com/JonChesterfield/dynamic_cpp_talk.git

October 26, 2016