

NIM, a tool for generating C code

DysFunctional Programmers

Jordan Hrycaj

<jordan@mjh-it.co.uk>

26/01/2021 (*previously Camsec 26/01/2017*)



Overview

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with NIM

The Language
NIM Workflow
Examples

Summary

Code Generation Tools For C

My Problem: Embedded/Multi-Platform Compiler

Why I would stay with C

Observations after choosing NIM

Working with NIM

The Language

NIM Workflow

Examples

Motivation

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

- ▶ Support for operating systems
 - ▶ Windows, Posix (Linux, Darwin), BSD, embedded
- ▶ Support for architectures
 - ▶ Intel/AMD, ARM, probably GPU
- ▶ Write code once, use it everywhere
 - ▶ JVM not an option on small systems
 - ▶ support hardware (mm registers, mmu)

Checking out well known systems

Comparing some alternatives to C.

- ▶ C++
 - ▶ The industry standard, has evolved considerably
 - ▶ Still not happy with C++ (eg. memory management)
 - ▶ Older systems might be unsupported (as of C++11/14)
 - ▶ Bloated binaries (think of IoT)
- ▶ Alternative Rust
 - ▶ Modern, seems to tick most boxes (ref counts, no GC)
 - ▶ Feels a bit like designed-by-committee
 - ▶ Older systems are unsupported
- ▶ Alternative GO
 - ▶ Many good ideas
 - ▶ A bit more low-level (than Rust)
 - ▶ GCC/GO not available on Windows
- ▶ More examples ...

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Using plain/bare bone C

Why using C at all?

- ▶ Coding in C can be a pain
 - ▶ Bloated code (but small binaries)
 - ▶ ... like crawling when you could use a car
- ▶ Bare bone C++ is possible
 - ▶ eg. w/o traps/exceptions
 - ▶ highly dependent on target system compiler/linker
- ▶ ... but plain C
 - ▶ It is universally supported, profiling, optimisers etc.
 - ▶ Fall back strategies (older systems, missing features)
 - ▶ Small language (compared to C++), features in libraries
- ▶ ... so generate C code, practised already widely
 - ▶ CPP, code generators (bison, flex, re2c etc.)
 - ▶ C itself

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C

Choosing NIM

Working with NIM

The Language

NIM Workflow

Examples

Summary

Solution: Compile to C

Compilation: high level language \rightarrow C \rightarrow binary

- ▶ Wish list for a compiler: Must Have
 - ▶ Functional support, closures
 - ▶ Convincing memory management (GC, ref count)
- ▶ Important
 - ▶ Complex but clean data structures
 - ▶ Generic C support (eg. inline, FFI)
 - ▶ Cross-compiling made easy
 - ▶ Templates/Macros
- ▶ Optional
 - ▶ Multi threading support (actors, pools, etc.)
 - ▶ Built-in OO
- ▶ Remark: *I found some really interesting stuff but many compilers produce C++ code which is what I wanted to avoid.*

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Useful tools to generate C code

Plain C compilers

- ▶ Chicken Scheme (since 2000)
 - ▶ Compiler/interpreter
 - ▶ Rich library (items called “eggs”)
 - ▶ Small runtime library
- ▶ Vala OO compiler (since 2006)
 - ▶ C# like programming language
 - ▶ GLib objects (can do without on a subset of features)
 - ▶ GLib seems to be portable but is bloated and big
- ▶ NIM (formerly Nimrod, since 2008)
 - ▶ Imperative, statically typed, functional
 - ▶ Influenced by Ada, C++, Lisp, C#, etc.
 - ▶ AST exposed for meta/macro programming
 - ▶ Produces C code ready for target system
 - ▶ Small runtime library (unless GC is avoided)

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C

Choosing NIM

Working with NIM

The Language

NIM Workflow

Examples

Summary

I ended up choosing NIM

But Chicken would have been OK as well

NIM, a tool for
generating C
code

Jordan Hrycaj

► Advantages

- Seems to be used in some industry/business applications
- Expressive syntax feels more like scripting (Perl/Python)
- Simple but (sometimes too) powerful way of coding
- Generics (~C++ templates), templates (simple macros)
- Macros (AST programming, ~Lisp macros) for DSL
- Can also produce C++, Obj C, and JS (probably others)

► Caveats

- Still experimental version
- Set up by a *benevolent dictator* + crew of enthusiasts
- C coding experience needed for NIM to be most useful
- No backing funds like for Go, Rust

► Verdict after using it for several months

- Useful even if support stops
- Most features I need are available

Code Generation
Tools For C

My Problem:
Embedded / Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Hello World

NIM is elegant (I am not aware of another elegant language)

NIM, a tool for
generating C
code

Jordan Hrycaj

- ▶ The programme

```
echo "Hello World"
```

- ▶ MAIN calling a function

```
proc helloWorld() =  
  echo "Hello World"  
helloWorld()
```

- ▶ MAIN calling a function with optional argument

```
proc helloWorld(text = "Hello World") =  
  echo text  
"Hello People".helloWorld
```

- ▶ Note that the type of text is string – inferred by its default argument. A more complete way of stating the argument would be: `text:string="Hello World"`

- ▶ See <http://nim-by-example.github.io>

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Oddities/Niceties

Things that are different in NIM

- ▶ Symbol names (1st character case important)
 - ▶ `theSymbol`, `the_symbol`, `theEsymbol` are equal
 - ▶ `TheSymbol` and `theSymbol` are different
- ▶ Closure support often needs annotation
 - ▶ For C the pragma `{.procvar.}` usually works
 - ▶ Compiler needs to figure out for potential concurrency
 - ▶ Results in plain C (no run time lib needed)
- ▶ Many NIM features are available at compile time
 - ▶ Can process files to create complex static data
 - ▶ Functional filters and operators but no OO
- ▶ Sequence functions `head()` and `tail()` are missing

```
proc tail*[T](s: openArray[T]): seq[T] {.inline.} =  
  if 0 < s.len: (@s)[1 .. <s.len] else: @[]
```
- ▶ Only finite sequence types supported
 - ▶ No tail recursion for formally unbounded sequences
 - ▶ Sequence type ad-hoc extensible

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler

Stay with C

Choosing NIM

Working with
NIM

The Language

NIM Workflow

Examples

Summary

Speed/Time Comparisons

<http://arthurtw.github.io/2015/01/12/quick-comparison-nim-vs-rust.html>

Game of Life	Rust	Nim/boundChecks:on	n=30000
with map print	1x	1.75x / 1.87x	1x=3.33s
without map print	1x	1.15x / 1.72x	1x=0.78s

<http://togototo.wordpress.com/2013/08/23/benchmarks-round-two-parallel-go-rust-d-scala-and-nimrod/>

Lang	Compiler	Speed/s	%Fastest	Res.Mem/KiB
D	ldc2	0.812	116.38%	26,536
C++	clang++	0.945	100.00%	25,552
Nimrod	clang	0.980	96.43%	25,932
C++	g++	1.025	92.20%	25,532
Rust	rustc	1.109	85.21%	47,708
Go	6g	1.184	79.81%	30,768
C	clang	1.199	78.82%	25,796
Scala	scala	1.228	76.95%	72,960
Go	gccgo	2.710	34.87%	69,120

(excerpt)

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Code Generation

NIM, a tool for
generating C
code

Jordan Hrycaj

- ▶ Compile-and-run a programme with
 - ▶ | `nim c -r helloworld.nim`
- ▶ what happens in the background is
 - ▶ The compiler builds up an AST
 - ▶ several compiler passes
 - ▶ imported code libraries are merged into the AST
 - ▶ Depending on the target code – assume C for now
 - ▶ C code files are generated
 - ▶ placed into the `./nimcache` directory
 - ▶ one C source per imported library
 - ▶ The compiler starts a C compiler on the C sources
 - ▶ optimised for GCC, Clang, Vcc (fallback: tiny CC)
 - ▶ produces binary
 - ▶ The binary is started

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

NIM Tools

Besides Compiler

- ▶ Rudimentary REPL
 - ▶ Install: `nimble install nrpl`
- ▶ NIM embedded debugger
 - ▶ `endb`, outdated
- ▶ GDB
 - ▶ Compile NIM with line pragmas enabled
 - ▶ Works fine for experienced C coder
- ▶ `nim2c`
 - ▶ Convert C code to NIM code
 - ▶ Handy tool, needs manual post-processing

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

Simple Functional Stuff

code walk over `misc/nimsrc.nim`

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

- ▶ generic function: `tail()`
 - ▶ generic `T`: any data type, normally inferred (see tests)
 - ▶ `openArray`: ordered data items of all the same type
 - ▶ type inference: `.. else @[]`
 - ▶ pattern matching for C optimisation
 - ▶ `seq` and `openArray`
 - ▶ check generated code
- ▶ function: `cnfValue()`
 - ▶ input `cnfTable()`: `seq[]` of string pairs (AVP list)
 - ▶ add item that always matches: `concat(@(s, ""))`
 - ▶ filter out first match: `filterIt(it[0] == s)`
 - ▶ get first match: `head()`
 - ▶ extract pair from sequence: `[0]`, get value: `[1]`

Compile Time Coding

code walk over `misc/nimsrc.nim`

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

- ▶ `template: nimSrcFilename()`
 - ▶ compiler support: `instantiationInfo()`
 - ▶ info about code that invokes it, so it must be a macro/template
- ▶ functions: `cnfTable()` and `cnfValue()`
 - ▶ extract AVP list from C header `config.h`
 - ▶ from `autoconf` environment
 - ▶ all compile time: `slurp/staticRead`, `gorge/staticExec`
 - ▶ no OO support
 - ▶ compiler quits if `slurp()` fails

C Bindings

code walk over `zlib/zlib.nim`

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

- ▶ C struct `z_stream` vs. NIM object `TZStream`
 - ▶ NIM objects are GC controlled (tuples are not)
 - ▶ compatibility types `cstring`, `cint`, `cuint`, etc.
- ▶ verify descriptor mapping: `zstreamspecs.c`
 - ▶ cross compiling `i386/x64/Linux/Windows` etc.
 - ▶ `cstring`, `cint`, probably struct alignments vary
 - ▶ see `import/binding` in test section: `tZstreamSpecs()`
 - ▶ `doAssert()` validity of descriptor mapping
- ▶ Zlib part is compiled all first
 - ▶ using macros and compile time lists
 - ▶ note the compile time path separator `D`
 - ▶ rather than `/` operator or `DirSep`
 - ▶ when host/target systems differ (eg. `Posix/Windows`)

NIM Summary

NIM, a tool for
generating C
code

Jordan Hrycaj

Code Generation
Tools For C

My Problem:
Embedded/Multi-
Platform Compiler
Stay with C
Choosing NIM

Working with
NIM

The Language
NIM Workflow
Examples

Summary

- ▶ Produces code for target system/compiler
 - ▶ C/C++/ObjC, JS
 - ▶ Multi paradigm language
 - ▶ extensible, DSL, accessible AST
- ▶ Targeting C
 - ▶ Easy to interface C libraries
 - ▶ Supports C cross compiling
 - ▶ GDB aware for debugging
- ▶ Young language
 - ▶ Small (but not too small) user group
 - ▶ Documentation OK (possibly more examples needed)
 - ▶ No big sponsors