Nim & C

Reaching the stars by standing on the shoulders of giants

Peter Munch-Ellingsen, M.Sc

PMunch – peterme.net

@pmunch@snabelen.no

Building a language

What is Nim?

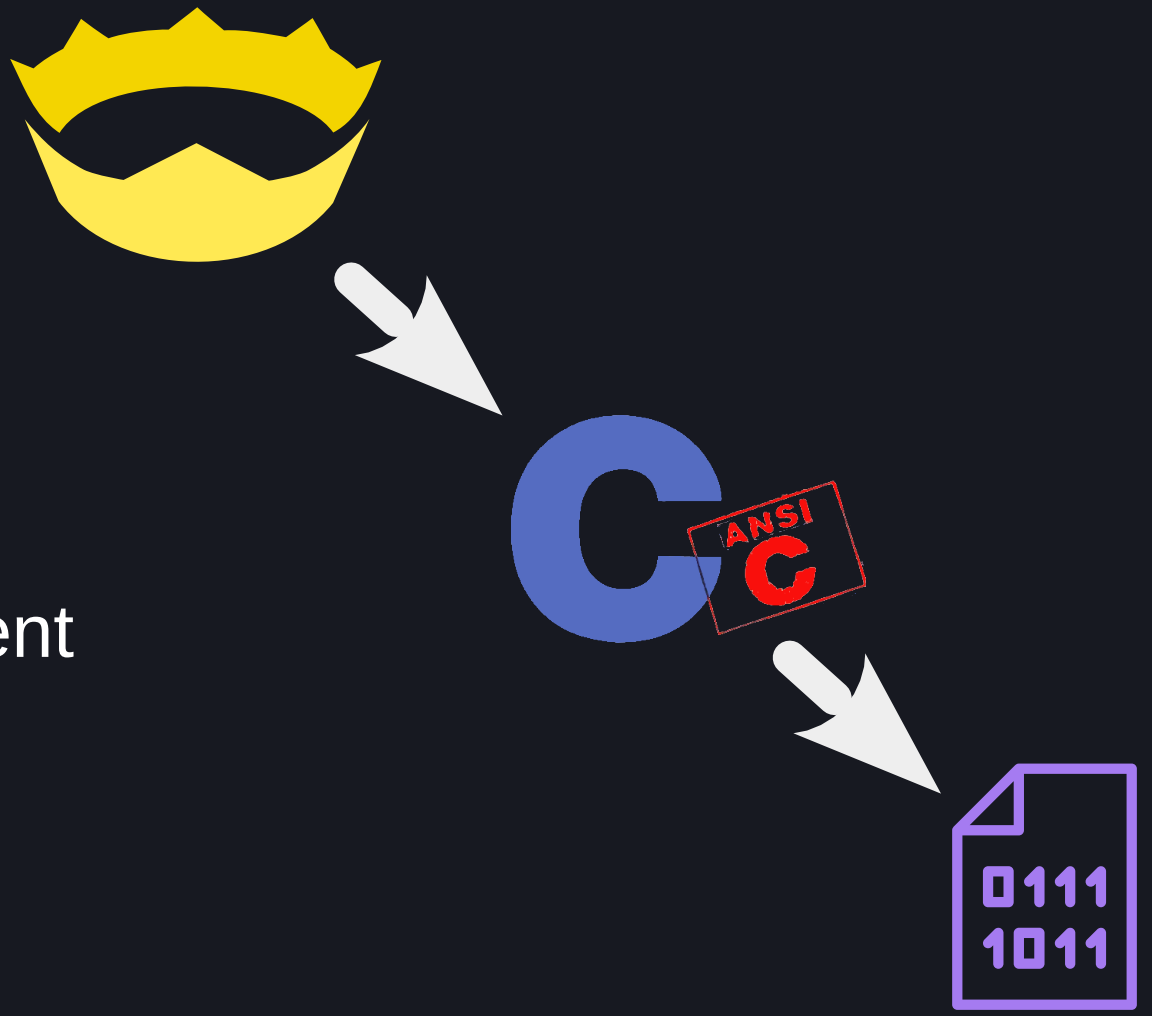
- Compiled
- Statically typed
- Flexible macro system
- Speed of C,
  ease of Python,
  flexibility of Perl

```
# Compute average line length
# From nim-lang.org
var
  sum = 0
  count = 0

for line in stdin.lines:
  sum += line.len
  count += 1

echo("Average line length: ",
     if count > 0: sum / count else: 0)
```

What is Nim?



- Compiles via C
- Easy FFI
- Hookable automatic memory management

Compiles to C?

```
# Compute average line length
var
  sum = 0
  count = 0

for line in stdin.lines:
  sum += line.len
  count += 1

echo("Average line length: ",
     if count > 0:
       sum / count else: 0)
```

```
while (1) {
    NIM_BOOL T6_;
    NI TM__EZNfNjUizwDC9c8XvLlPvow_2;
    NI TM__EZNfNjUizwDC9c8XvLlPvow_3;
    T6_ = (NIM_BOOL)0;
    T6_ = readLine__stdZsyncio_u286(((FILE*) (
    if (NIM_UNLIKELY(*nimErr_)) goto LA3_;
    if (!T6_) goto LA5;
    eqcopy___stdZassertions_u30((&line__lines_
    if (nimAddInt(sum__lines_u1, line__lines_u
    };
    sum__lines_u1 = (NI)(TM__EZNfNjUizwDC9c8Xv
    if (nimAddInt(count__lines_u2, ((NI)1), &T
    };
    count__lines_u2 = (NI)(TM__EZNfNjUizwDC9c8
} LA5: ;
```

The foundations

logic.c:

```c
int addTwoIntegers(int a, int b)
{
    return a + b;
}
```

calculator.nim:

```nim
{.compile: "logic.c".}
proc addTwoIntegers(a, b: cint): cint {.importc.}

when isMainModule:
  echo addTwoIntegers(3, 7)
```

The automatic approach

The automatic approach

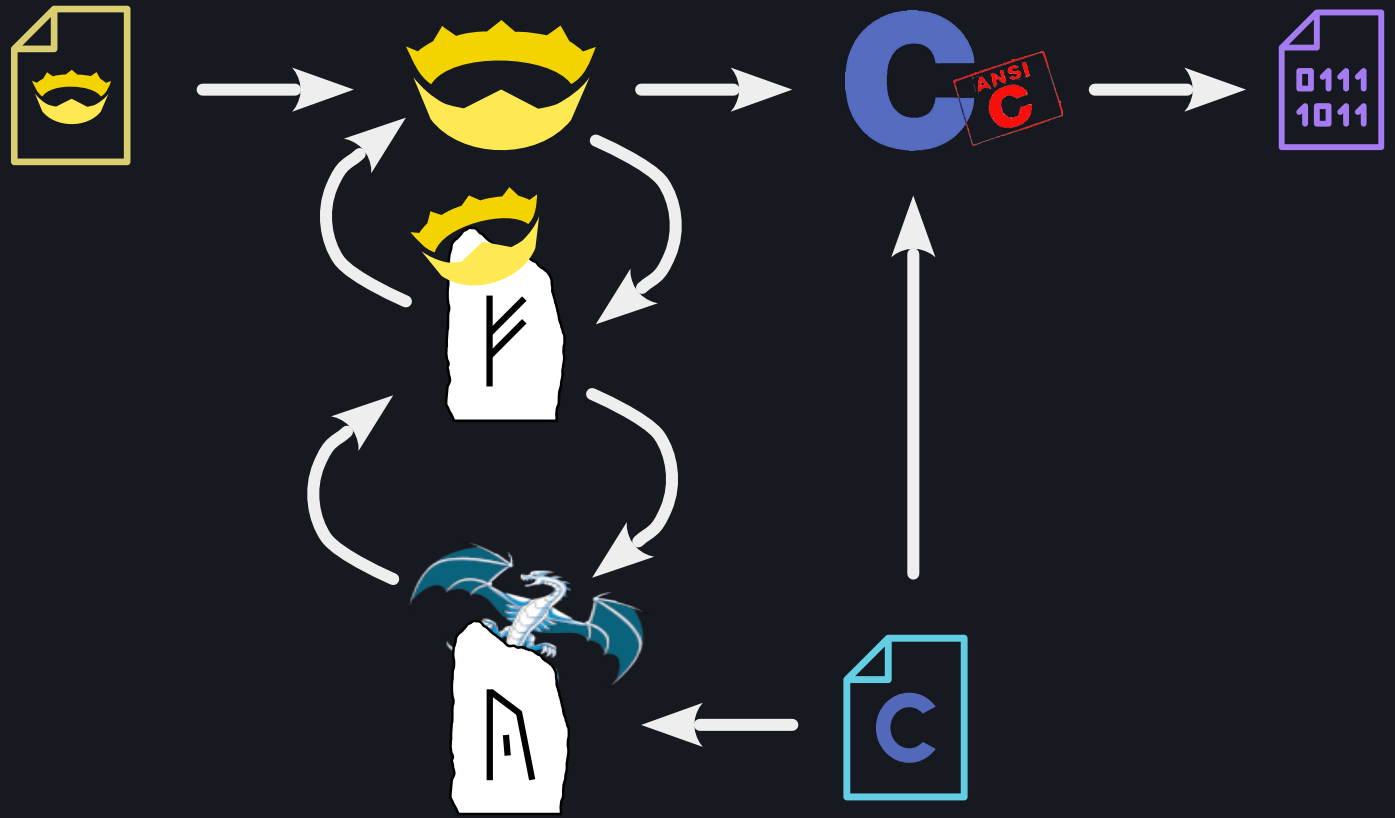
The automatic approach

```
import futhark, os

{.passL: currentSourcePath.parentDir() & "/libmapm.a".}

importc:
  path "./mapm"
  "m_apm.h"
  rename M_APM, M_APM_INTERNAL
```

The automatic approach



The automatic approach

```
type
  struct_M_APM_struct* {.pure, inheritable, bycopy.} = object
    m_apm_data*: ptr UCHAR   ## Generated based on /tmp/mapm/m_apm.h:173:9
    m_apm_id*: clong
    m_apm_refcount*: cint
    [...]

proc m_apm_arcsin*(a0: Mapminternal; a1: cint; a2: Mapminternal): void {.cdecl,
    importc: "m_apm_arcsin".}
proc m_apm_arccos*(a0: Mapminternal; a1: cint; a2: Mapminternal): void {.cdecl,
    importc: "m_apm_arccos".}

var MM_Two* {.importc: "MM_Two".}: Mapminternal
var MM_Three* {.importc: "MM_Three".}: Mapminternal
var MM_Four* {.importc: "MM_Four".}: Mapminternal

proc m_apm_init*(): Mapminternal {.cdecl, importc: "m_apm_init".}
proc m_apm_free*(a0: Mapminternal): void {.cdecl, importc: "m_apm_free".}
```

The automatic approach

```
import mapm

var
  x = m_apm_init()
  y = m_apm_init()
  r = m_apm_init()
m_apm_set_long(x, 10.clong)
m_apm_set_double(x, 6.4e+2)
m_apm_divide(r, 2.cint, x, y)
var s = newString(10)
m_apm_to_fixpt_string(cast[cstring](s[0].addr), -1, r)
echo s
```

The automatic approach

```
import mapm

var x = initMapm(10) # Initialize explicitly
echo x / 6.4e+2'm # Or through the 'm format specifier

let y = x + 5.342'm * MM_Pi # Use like any other number
```

Limitations

+ Handles pretty much any C code
+ Very automated process with clean fallbacks
~ Agnostic to C linking
~ Method easily portable to C++
- Still not great on function style macros
- Only wraps C code, doesn't "Nimify" it

Nim & C

Reaching the stars by standing on the shoulders of giants

Peter Munch-Ellingsen, M.Sc

PMunch – peterme.net

@pmunch@snabelen.no