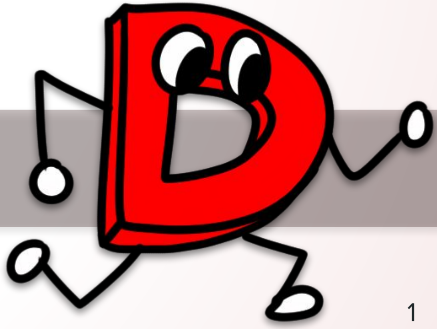
The Jack of all trades

Dennis Korpel - DConf 2022



Automatic memory management makes for safe, simple, and robust code. D also supports scoped resource management (aka the RAII idiom) and **scope** statements for deterministic transactional code that is easy to write and read. Show example ▾

Built-in linear and associative arrays, slices, and ranges make daily programming simple and pleasant for tasks, both small and large. Show example ▾

Read Fast

The best paradigm is to not impose something at the expense of others. D offers classic polymorphism, value semantics, functional style, generics, generative programming, contract programming, and more—all harmoniously integrated. Show example ▾

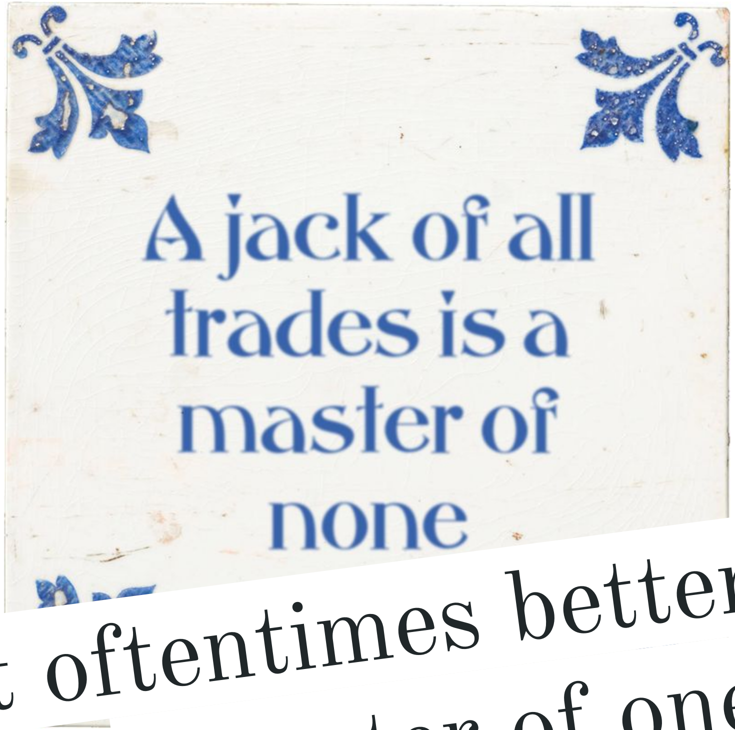
D offers an innovative approach to concurrency, featuring true immutable data, message passing, no sharing by default, and controlled mutable sharing across threads. Read more.

From simple scripts to large projects, D has the breadth to scale with any application's needs: unit testing, information hiding, refined modularity, fast compilation, precise interfaces. Read more.

Run Fast

D compiles naturally to efficient native code.

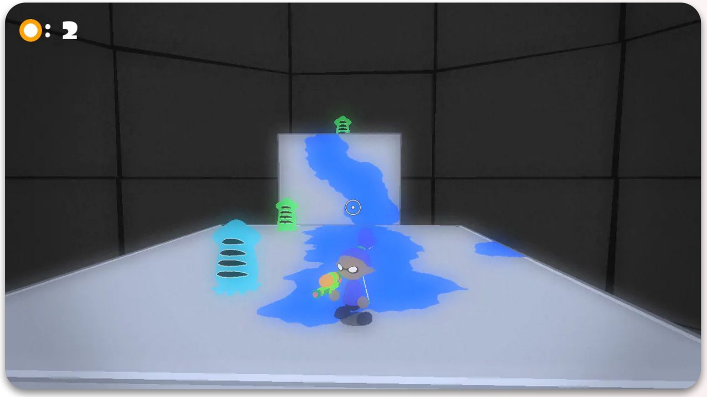
D is designed such that most "obvious" code is fast *and* safe. On occasion a function might need to escape the confines of type safety for ultimate speed and control. For such rare cases D offers native pointers, type casts, access to any C function without any intervening translation, manual memory management, custom allocators and even inline assembly code. Show example ▾


A jack of all trades is a master of none

but oftentimes better than a master of one

My Language usage

Bestuur de bal met de pijltjestoetsen - druk op R om te herstarten - druk op Esc om terug te gaan naar het menu



GAME MAKER

2008 2015 2022

Game Maker Language (GML)

- All code does something

- Compiles to single .exe

GML

```
if (keyboard_pressed(vk_up) and on_ground)
    {
    vspeed = -10
    sound_play(snd_jump)
    sprite_index = spr_player_jump
    }
```

My Language usage

- BSc. Computer Science at Delft University of Techology

Java
C / C++
x86 / MIPS asm
Python
Typescript
Prolog
Coq
Scala
MiniZinc
Julia
LUA
C#

GAME MAKER

2008 2015 2018 2022

Java

- All code does something... No

Java

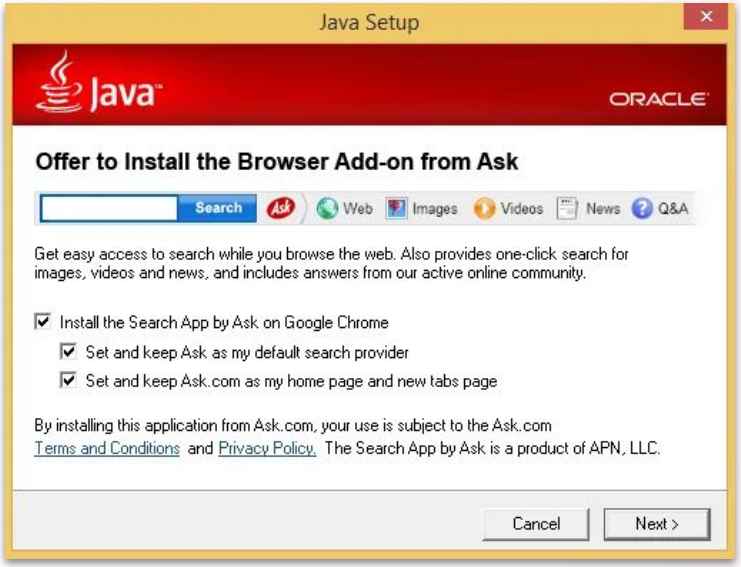
```
public class StubFactoryFactoryProxyImpl extends StubFactoryFactoryDynamicBase
{
    public PresentationManager.StubFactory makeDynamicStubFactory(
          PresentationManager pm, PresentationManager.ClassData classData,
          ClassLoader classLoader )
    {
          return new StubFactoryProxyImpl( classData, classLoader ) ;
    }
}
```

http://harmful.cat-v.org/Blog/2009/08/05/0-java-factory/

Java

- All code does something... No

- Compiles to `.jar`, requires setup



Offer to Install the Browser Add-on from Ask

Get easy access to search while you browse the web. Also provides one-click search for images, videos and news, and includes answers from our active online community.

[✓] Install the Search App by Ask on Google Chrome

[✓] Set and keep Ask as my default search provider

[✓] Set and keep Ask.com as my home page and new tabs page

By installing this application from Ask.com, your use is subject to the Ask.com Terms and Conditions and Privacy Policy. The Search App by Ask is a product of APN, LLC.

C++

- "C++ is my favorite language once I learn it"

- Segfaults instead of Exception traces

- Still had to ship `glfw3.dll`

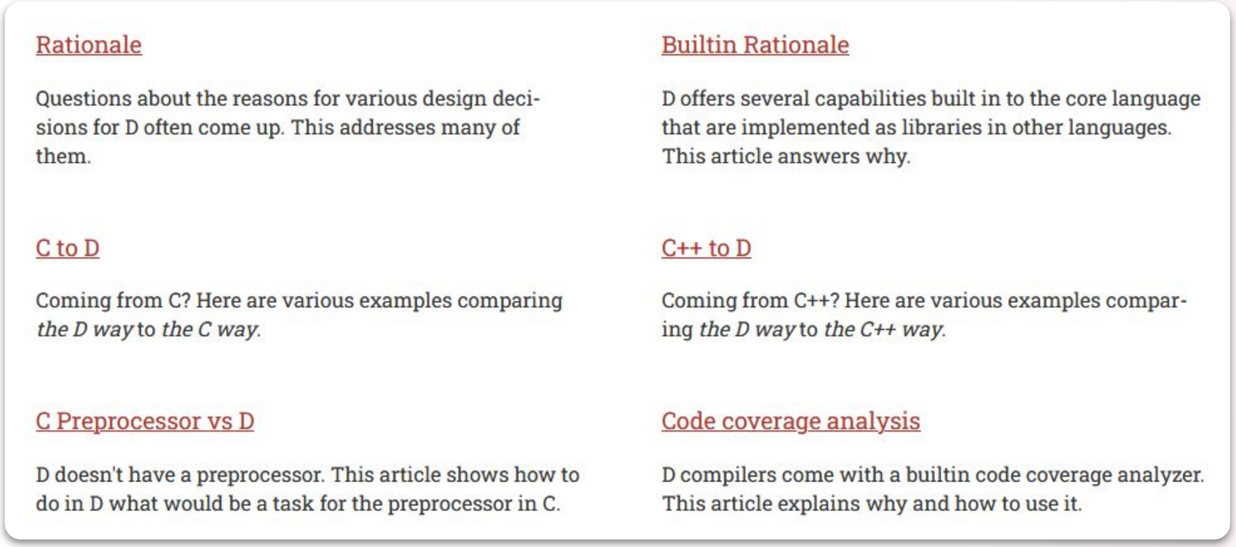
- Discovered D on benchmark site

- "They made a sequel to C/C++?"

Language	Relative runtime
C	1.0
C++	1.0
D	1.1
Java	1.8

D

- Good rationale
- Automatic boilerplate
- Compiles to `.exe`



Rationale

Questions about the reasons for various design decisions for D often come up. This addresses many of them.

Builtin Rationale

D offers several capabilities built in to the core language that are implemented as libraries in other languages. This article answers why.

C to D

Coming from C? Here are various examples comparing *the D way* to *the C way*.

C++ to D

Coming from C++? Here are various examples comparing *the D way* to *the C++ way*.

C Preprocessor vs D

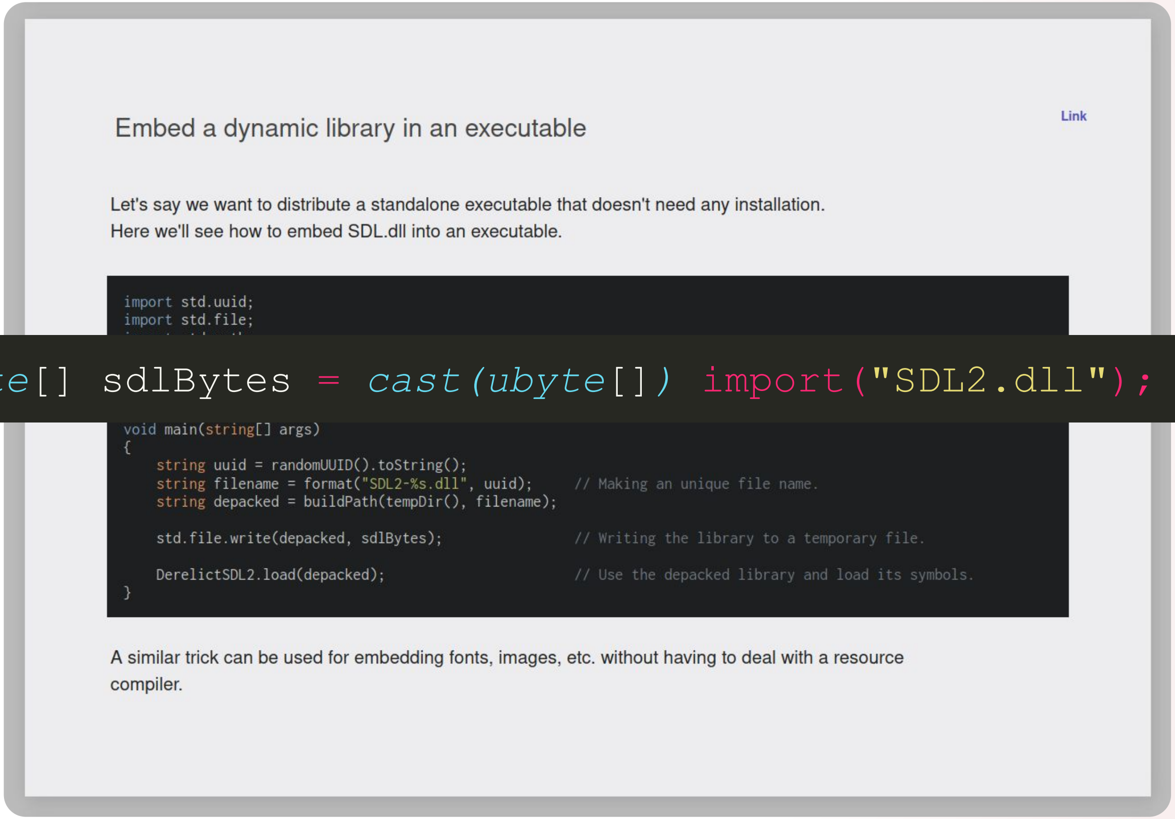
D doesn't have a preprocessor. This article shows how to do in D what would be a task for the preprocessor in C.

Code coverage analysis

D compilers come with a builtin code coverage analyzer. This article explains why and how to use it.

D

- Good rationale
- Automatic boilerplate
- Compiles to `.exe`



Embed a dynamic library in an executable

Link

Let's say we want to distribute a standalone executable that doesn't need any installation.
Here we'll see how to embed SDL.dll into an executable.

```
import std.uuid;
import std.file;
```

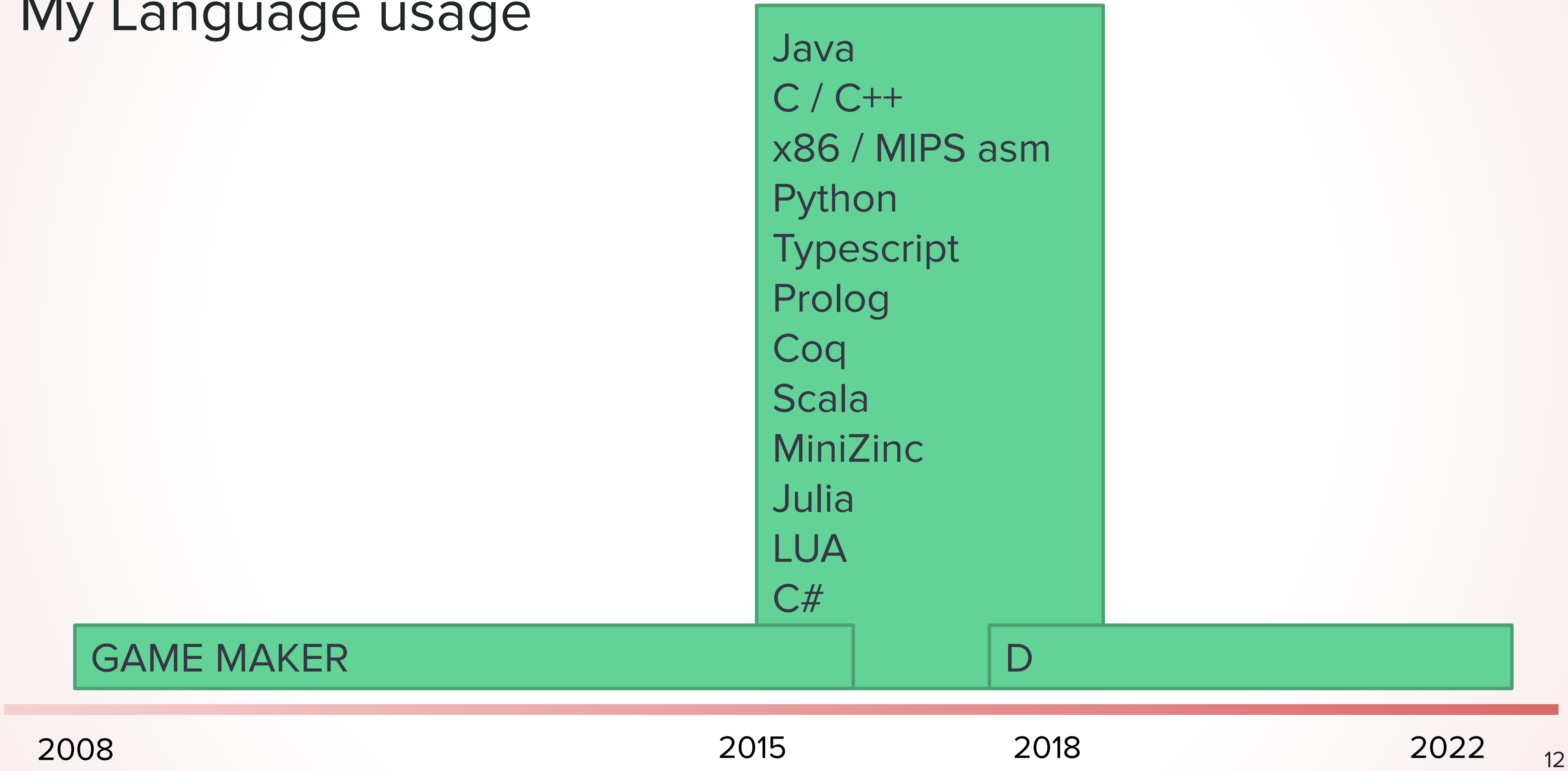
```
ubyte[] sdlBytes = cast(ubyte[]) import("SDL2.dll");
```

```
void main(string[] args)
{
    string uuid = randomUUID().toString();
    string filename = format("SDL2-%s.dll", uuid);     // Making an unique file name.
    string depacked = buildPath(tempDir(), filename);

    std.file.write(depacked, sdlBytes);                // Writing the library to a temporary file.

    DerelictSDL2.load(depacked);                       // Use the depacked library and load its symbols.
}
```

A similar trick can be used for embedding fonts, images, etc. without having to deal with a resource compiler.

https://github.com/p0nce/d-idioms/commit/0661563ccd712aa0d91a2110aa480823a8ba9cc6

My Language usage



Using D for *evertyhing*

- Lots of hobby projects in D
- Why not use specialized languages?
- Complexity in big language

OpenGL app

MIPS assembler

ELF linker

Scripts

Computer algebra system

Software synthesizer

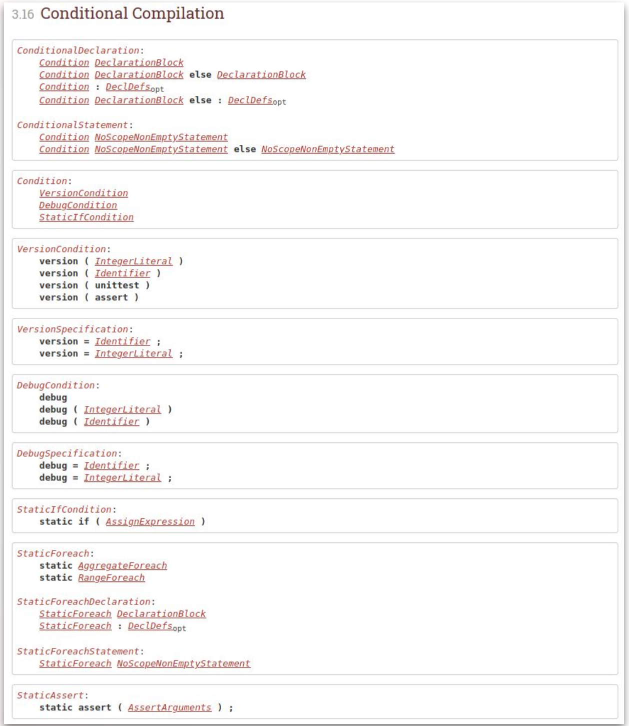
C to D translator

Ultimate tic-tac-toe game

Codingame challenges

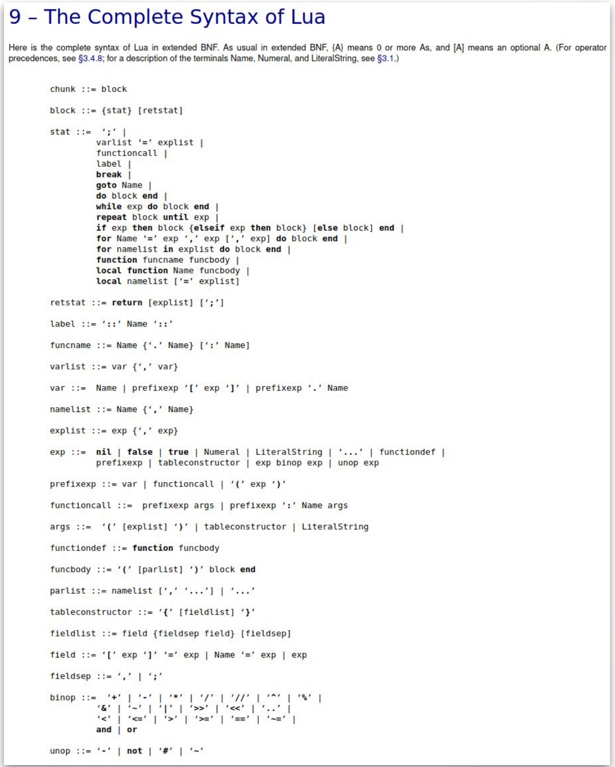
Chess game

Complexity in the spec



3.16 Conditional Compilation

```
ConditionalDeclaration:
    Condition DeclarationBlock
    Condition DeclarationBlock else DeclarationBlock
    Condition : DeclDefsopt
    Condition DeclarationBlock else : DeclDefsopt

ConditionalStatement:
    Condition NoScopeNonEmptyStatement
    Condition NoScopeNonEmptyStatement else NoScopeNonEmptyStatement

Condition:
    VersionCondition
    DebugCondition
    StaticIfCondition

VersionCondition:
    version ( IntegerLiteral )
    version ( Identifier )
    version ( unittest )
    version ( assert )

VersionSpecification:
    version = Identifier ;
    version = IntegerLiteral ;

DebugCondition:
    debug
    debug ( IntegerLiteral )
    debug ( Identifier )

DebugSpecification:
    debug = Identifier ;
    debug = IntegerLiteral ;

StaticIfCondition:
    static if ( AssignExpression )

StaticForeach:
    static AggregateForeach
    static RangeForeach

StaticForeachDeclaration:
    StaticForeach DeclarationBlock
    StaticForeach : DeclDefsopt

StaticForeachStatement:
    StaticForeach NoScopeNonEmptyStatement

StaticAssert:
    static assert ( AssertArguments ) ;
```



9 – The Complete Syntax of Lua

Here is the complete syntax of Lua in extended BNF. As usual in extended BNF, {A} means 0 or more As, and [A] means an optional A. (For operator precedences, see §3.4.8; for a description of the terminals Name, Numeral, and LiteralString, see §3.1.)

```
chunk ::= block

block ::= {stat} [retstat]

stat ::=  ';' |
         varlist '=' explist |
         functioncall |
         label |
         break |
         goto Name |
         do block end |
         while exp do block end |
         repeat block until exp |
         if exp then block {elseif exp then block} [else block] end |
         for Name '=' exp ',' exp [',' exp] do block end |
         for namelist in explist do block end |
         function funcname funcbody |
         local function Name funcbody |
         local namelist ['=' explist]

retstat ::= return [explist] [';']

label ::= '::' Name '::'

funcname ::= Name {'.' Name} [':' Name]

varlist ::= var {',' var}

var ::=  Name | prefixexp '[' exp ']' | prefixexp '.' Name

namelist ::= Name {',' Name}

explist ::= exp {',' exp}

exp ::=  nil | false | true | Numeral | LiteralString | '...' | functiondef |
         prefixexp | tableconstructor | exp binop exp | unop exp

prefixexp ::= var | functioncall | '(' exp ')'

functioncall ::=  prefixexp args | prefixexp ':' Name args

args ::=  '(' [explist] ')' | tableconstructor | LiteralString

functiondef ::= function funcbody

funcbody ::= '(' [parlist] ')' block end

parlist ::= namelist [',' '...'] | '...'

tableconstructor ::= '{' [fieldlist] '}'

fieldlist ::= field {fieldsep field} [fieldsep]

field ::= '[' exp ']' '=' exp | Name '=' exp | exp

fieldsep ::= ',' | ';'

binop ::=  '+' | '-' | '*' | '/' | '//' | '^' | '%' |
         '&' | '~' | '|' | '>>' | '<<' | '..' |
         '<' | '<=' | '>' | '>=' | '==' | '~=' |
         and | or

unop ::= '-' | not | '#' | '~'
```

https://dlang.org/spec/grammar.html https://www.lua.org/manual/5.3/manual.html

Complexity for the user

- Aggregate two integers?

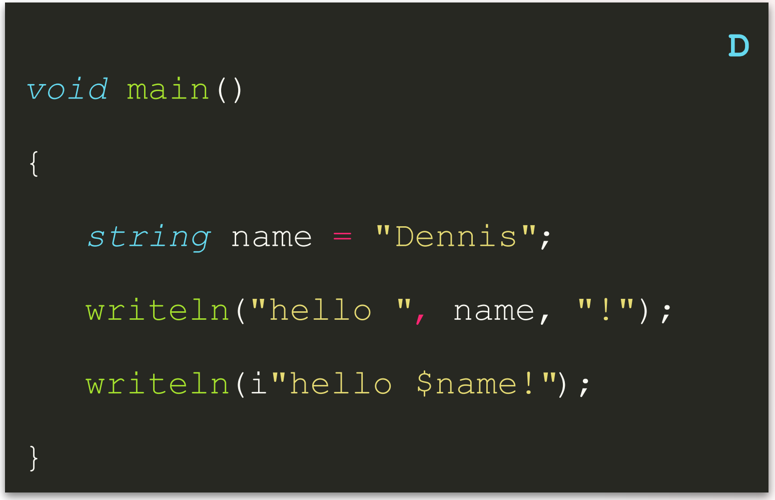
C: `struct`   Java: `class`   Lua: table

D:

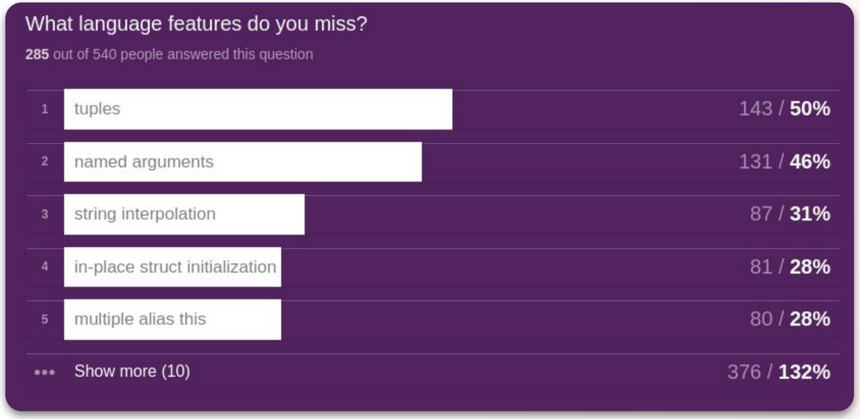
```
D
struct Pair { int x; int y; }
class  Pair { int x; int y; }
alias Pair = Tuple!(int, int);
alias Pair = int[2];
```

Complexity in design

- String interpolation
- Feature in C#, JavaScript, Python...
- Just add it to D?



```d
void main()
{
    string name = "Dennis";
    writeln("hello ", name, "!");
    writeln(i"hello $name!");
}
```



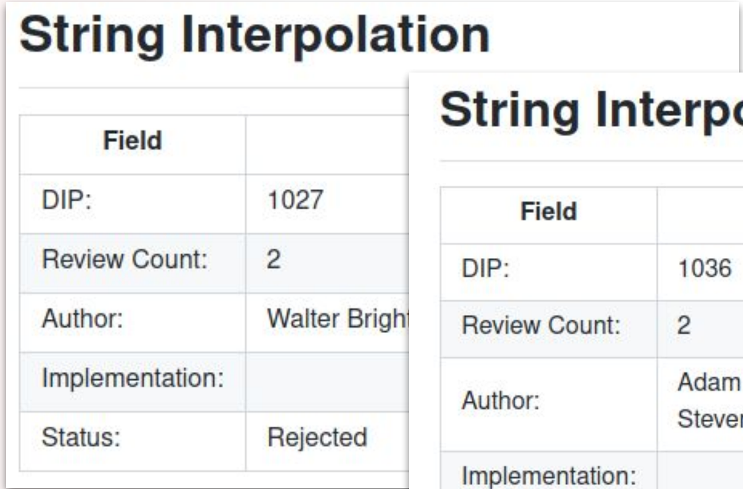
What language features do you miss?

285 out of 540 people answered this question

1	tuples	143 / 50%
2	named arguments	131 / 46%
3	string interpolation	87 / 31%
4	in-place struct initialization	81 / 28%
5	multiple alias this	80 / 28%
•••	Show more (10)	376 / 132%

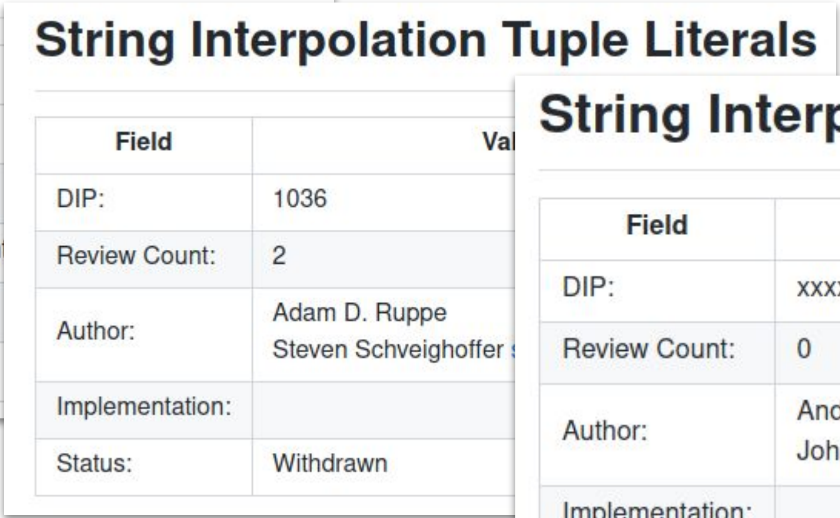
https://rawgit.com/wilzbach/state-of-d/master/report.html

Complexity in design



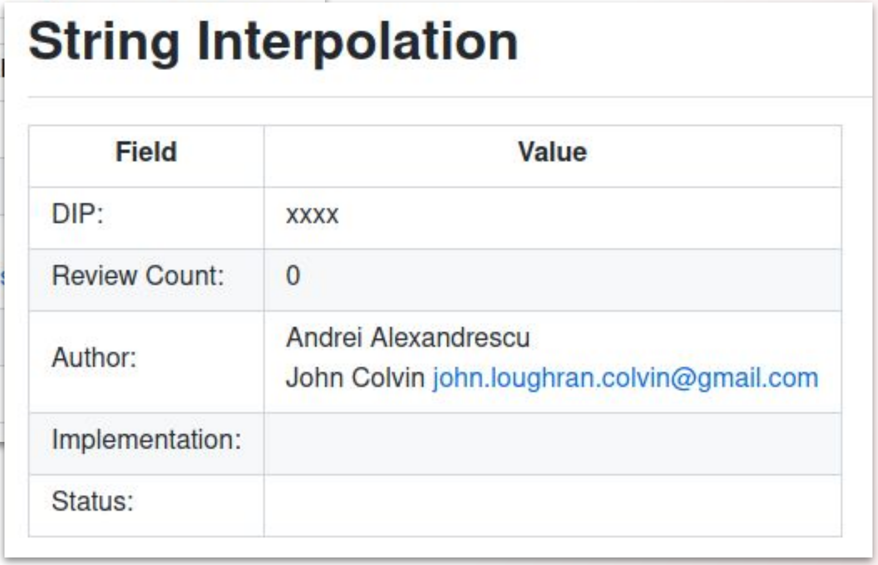
String Interpolation

Field	
DIP:	1027
Review Count:	2
Author:	Walter Brigh
Implementation:	
Status:	Rejected



String Interpolation Tuple Literals

Field	Val
DIP:	1036
Review Count:	2
Author:	Adam D. Ruppe Steven Schveighoffer
Implementation:	
Status:	Withdrawn



String Interpolation

Field	Value
DIP:	xxxx
Review Count:	0
Author:	Andrei Alexandrescu John Colvin john.loughran.colvin@gmail.com
Implementation:	
Status:	

https://dlang.org/dips/1027 https://dlang.org/dips/1036 https://github.com/John-Colvin/YAIDIP

String interpolation design challenges

- `writefln!i"$x"`
- `@nogc`
- `mixin(i"void $f() {}");`
- `printf(i"$x");`
- simple, easy to use

Better to mix languages?

High-level

Low-level

THE C PROGRAMMING LANGUAGE

Scripting

Lua

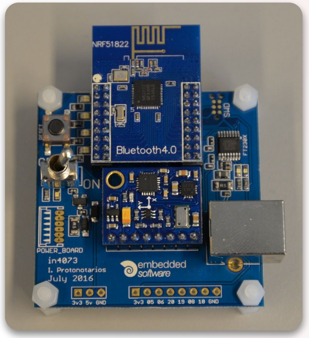
Application

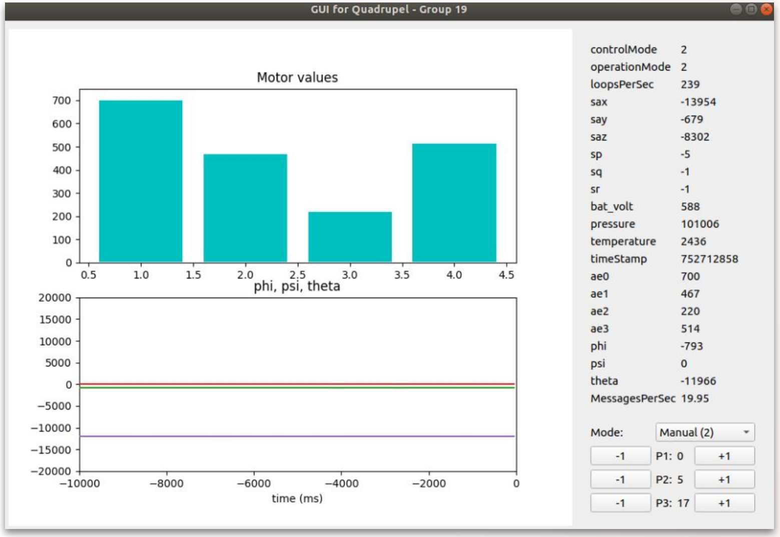
C#

High level Python

- Embedded code written in C

- GUI in Python





controlMode	2
operationMode	2
loopsPerSec	239
sax	-13954
say	-679
saz	-8302
sp	-5
sq	-1
sr	-1
bat_volt	588
pressure	101006
temperature	2436
timeStamp	752712858
ae0	700
ae1	467
ae2	220
ae3	514
phi	-793
psi	0
theta	-11966
MessagesPerSec	19.95

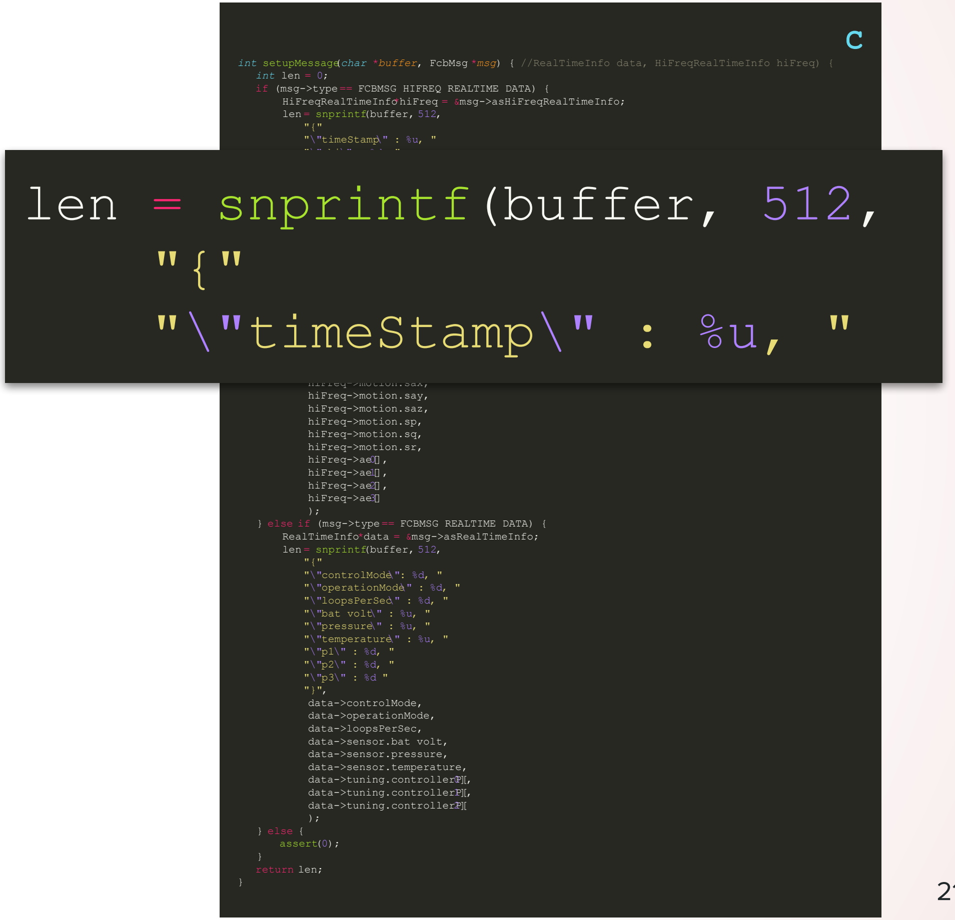
Mode: Manual (2)

-1	P1: 0	+1
-1	P2: 5	+1
-1	P3: 17	+1

https://www.st.ewi.tudelft.nl/koen/cs4140/

High level Python

- Embedded code written in C
- GUI in Python
- JSON sent over a socket
- Hard to maintain



C

```c
int setupMessage(char *buffer, FcbMsg *msg) { //RealTimeInfo data, HiFreqRealTimeInfo hiFreq) {
   int len = 0;
   if (msg->type == FCBMSG HIFREQ REALTIME DATA) {
       HiFreqRealTimeInfo*hiFreq = &msg->asHiFreqRealTimeInfo;
       len = snprintf(buffer, 512,
           "{"
           "\"timeStamp\" : %u, "
```

```c
len = snprintf(buffer, 512,
    "{"
    "\"timeStamp\" : %u, "
```

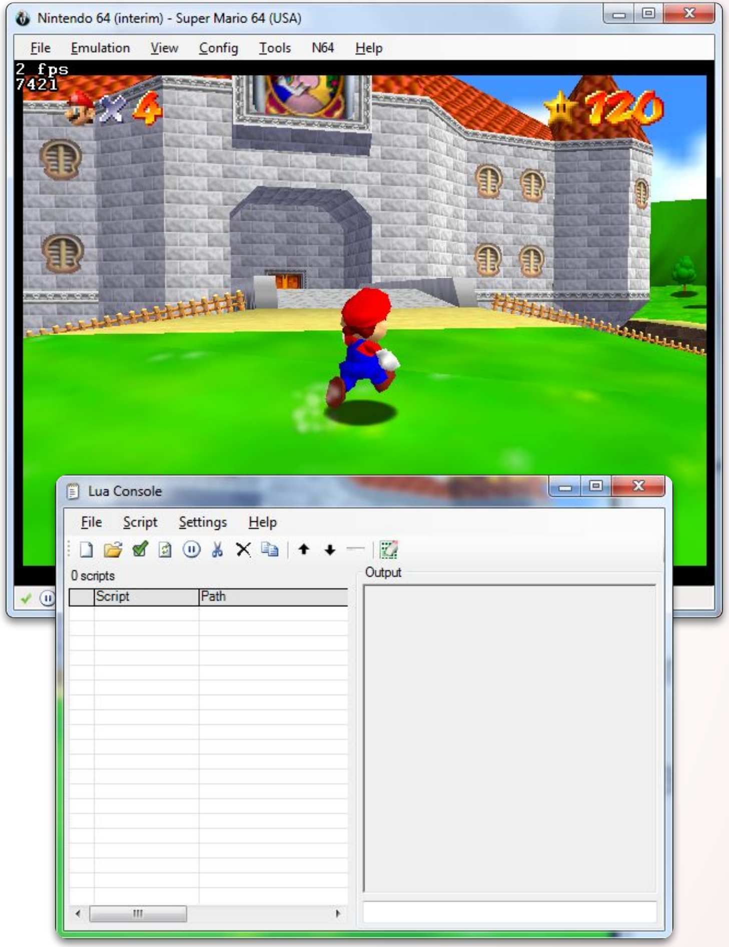
```c
               hiFreq->motion.say,
               hiFreq->motion.saz,
               hiFreq->motion.sp,
               hiFreq->motion.sq,
               hiFreq->motion.sr,
               hiFreq->ae0[],
               hiFreq->ae1[],
               hiFreq->ae2[],
               hiFreq->ae3[]
               );
   } else if (msg->type == FCBMSG REALTIME DATA) {
       RealTimeInfo*data = &msg->asRealTimeInfo;
       len = snprintf(buffer, 512,
           "{"
           "\"controlMode\": %d, "
           "\"operationMode\" : %d, "
           "\"loopsPerSec\" : %d, "
           "\"bat volt\" : %u, "
           "\"pressure\" : %u, "
           "\"temperature\" : %u, "
           "\"p1\" : %d, "
           "\"p2\" : %d, "
           "\"p3\" : %d "
           "}",
            data->controlMode,
            data->operationMode,
            data->loopsPerSec,
            data->sensor.bat volt,
            data->sensor.pressure,
            data->sensor.temperature,
            data->tuning.controller0P][,
            data->tuning.controller1P][,
            data->tuning.controller2P][
            );
   } else {
       assert(0);
   }
   return len;
}
```

LUA scripting

- Super Mario 64 (1996)

- "How did they code it?"

- Emulator with Lua scripting

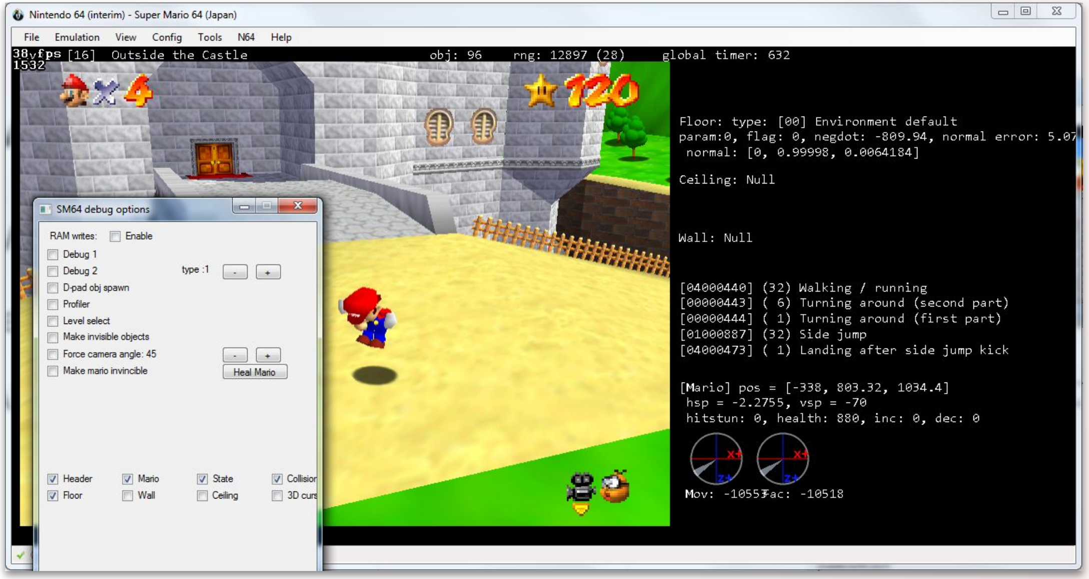

Nintendo 64 (interim) - Super Mario 64 (USA)
File Emulation View Config Tools N64 Help
2 fps
7421
4
120
Lua Console
File Script Settings Help
0 scripts
Script
Path
Output

Increasing complexity

- From small script to 5000 LOC

  Suddenly!

- Runtime `nil` errors


Nintendo 64 (interim) - Super Mario 64 (Japan)
File Emulation View Config Tools N64 Help
SM64 debug options
Heal Mario

```
38vfps [16]  Outside the Castle                obj: 96    rng: 12897 (28)     global timer: 632
1532

Floor: type: [00] Environment default
param:0, flag: 0, negdot: -809.94, normal error: 5.07
 normal: [0, 0.99998, 0.0064184]

Ceiling: Null

Wall: Null

[04000440] (32) Walking / running
[00000443] ( 6) Turning around (second part)
[00000444] ( 1) Turning around (first part)
[01000887] (32) Side jump
[04000473] ( 1) Landing after side jump kick

[Mario] pos = [-338, 803.32, 1034.4]
 hsp = -2.2755, vsp = -70
 hitstun: 0, health: 880, inc: 0, dec: 0

Mov: -10553Fac: -10518
```

RAM writes: [ ] Enable
[ ] Debug 1
[ ] Debug 2 type :1
[ ] D-pad obj spawn
[ ] Profiler
[ ] Level select
[ ] Make invisible objects
[ ] Force camera angle: 45
[ ] Make mario invincible

[✓] Header [✓] Mario [✓] State [✓] Collision
[✓] Floor [ ] Wall [ ] Ceiling [ ] 3D curs

Type systems

- "Static typing" in Lua

```lua
                                                                                                                  Lua
local function checkTypes()
   for typeName, typeObj in pairs( typeTable) do
       assert(type(typeObj._name) == "string", "Type "..typeName.." is not a string but a
"..type(typeObj. name))
       if (typeObj. prim == false and typeObj._dict ~= nil) then
           for k, v in pairs(typeObj. dict) do
               local tt =  typeTable[v.dataType]
               assert(tt ~= nil, "In struct type '"..typeName.."' key ["..k.."] has unknown type "..v.dataType)
               assert( (v.dataType ~= "pointer" and v.dataType ~= "array") or (v.parameter ~= nil) ,
                   "In dictionary of "..typeName.." in "..v.dataType.." entry "..k.." has no parameter." )
           end
       end

       local t0 = type(typeObj. readFunc)
       assert(t0 == "function" or t0 == "userdata", "Read function of "..typeObj._name.." is of type "..t0)
       t0 = type(typeObj. writeFunc)
       assert(t0 == "function" or t0 == "userdata", "Write function of "..typeObj._name.." is of type "..t0)
   end
end
```

Type systems

- "Static typing" in Lua

- "Dynamic typing" in D

D

```d
T max(T)(T x, T y)
{
    return x > y ? x : y;
}

const x = max(10, 20);
```

Seamless data

- One slice / dynamic array type

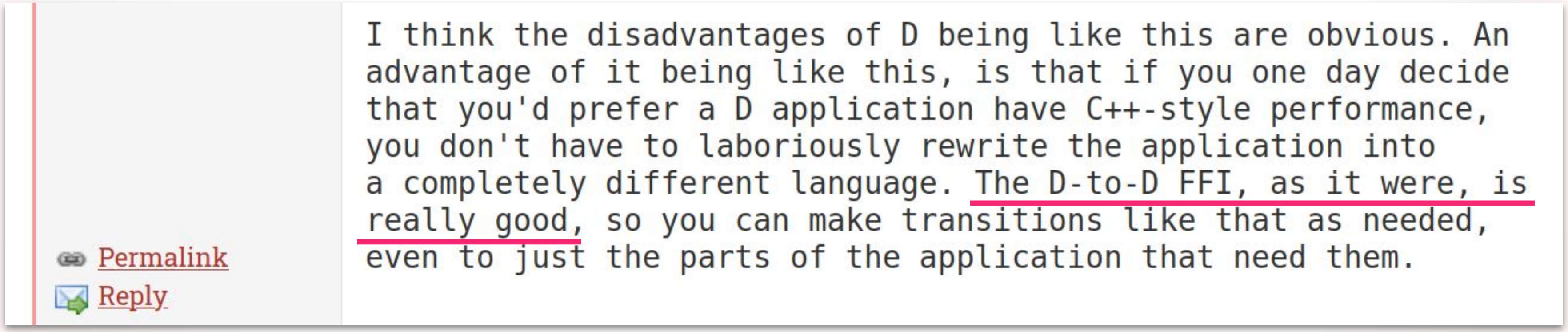
D

```d
void lowLevel()
{
    import core.stdc.stdio : snprintf;
    char[8] buf;
    const n = snprintf(buf.ptr, buf.length, "%d", 99);
    highLevel(buf[0 .. n]);
}

void highLevel(char[] a)
{
    import std.stdio : writeln;
    a ~= " bottles";
    writeln(a);
}
```

Seamless data

- One slice / dynamic array type

- Good D-to-D FFI



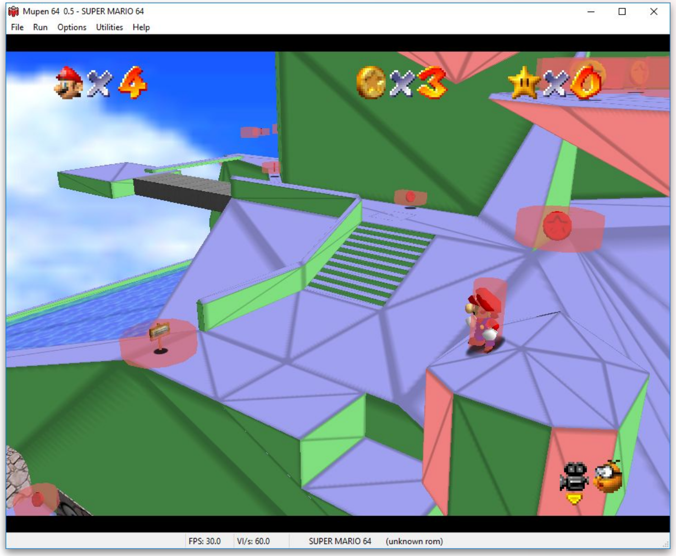
I think the disadvantages of D being like this are obvious. An advantage of it being like this, is that if you one day decide that you'd prefer a D application have C++-style performance, you don't have to laboriously rewrite the application into a completely different language. The D-to-D FFI, as it were, is really good, so you can make transitions like that as needed, even to just the parts of the application that need them.

D code

in a Nintendo 64 emulator

Assembly hacking

- Collision viewer
- Lua API too limited / slow
- Inject MIPS assembly
- Bugs


Mupen 64 0.5 - SUPER MARIO 64
File Run Options Utilities Help
×4
×3
×0
FPS: 30.0 VI/s: 60.0 SUPER MARIO 64 (unknown rom)

MIPS Assembly

```
LUI AT, 0x8039
LW T0, 0xEE9C (AT)
SLL T1, S0, 2      // 48*S0 = (4*S0-S0)*16
SUB T1, T1, S0
SLL T1, T1, 4
ADDU S5, T0, T1    // S5 = tri ptr = *38EE9C + 48*i
```

https://gist.github.com/dkorpel/dc7c435bf937fe886b67bfb51b7ec43a

Assembly hacking

- Write my own language?
- LDC has `-march=mips -mcpu=mips3`
- And `-output-s` flag
- How to resolve labels?

MIPS Assembly

```
$CPI2 1:
    .4byte   0x45266000
    .section     .text.hmain,"ax",@progbits
    .globl   hmain
    .p2align     3
    .type    hmain,@function
    .set     nomicromips
    .set     nomips16
    .ent     hmain
hmain:
    .cfi startproc
    .frame   $sp, 32,$ra
    .mask    0x80030000,-4
    .fmask   0x00000000,0
    .set     noreorder
    .set     nomacro
    .set     noat
    lui  $2, %hi( gp disp)
    addiu    $2, $2, %lo( gp_disp)
    addiu    $sp, $sp, - 32
    .cfi def cfa offset  32
    sw   $ra,  28($sp)
    sw   $17,  24($sp)
    sw   $16,  20($sp)
    .cfi offset  31, -4
    .cfi offset  17, -8
    .cfi offset  16, -12
    addu     $16, $2, $25
    lw   $2, %got(isInitialized)($16)
    lbu  $1,  0($2)
    bnez     $1, $BB2_2
    nop
    addiu    $1, $zero,  1
    sb   $1,  0($2)
```

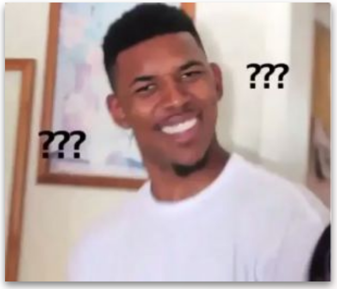
Linkers

- Relocation table
- Don't need assembler
- ELF binary format

"A linker is a very stupid, pedestrian, straightforward program. (...) The tedium in writing a linker is usually all about decoding and generating the usually ridiculously overcomplicated file formats" - Walter Bright

SHT_MIPS_ABIFLAGS

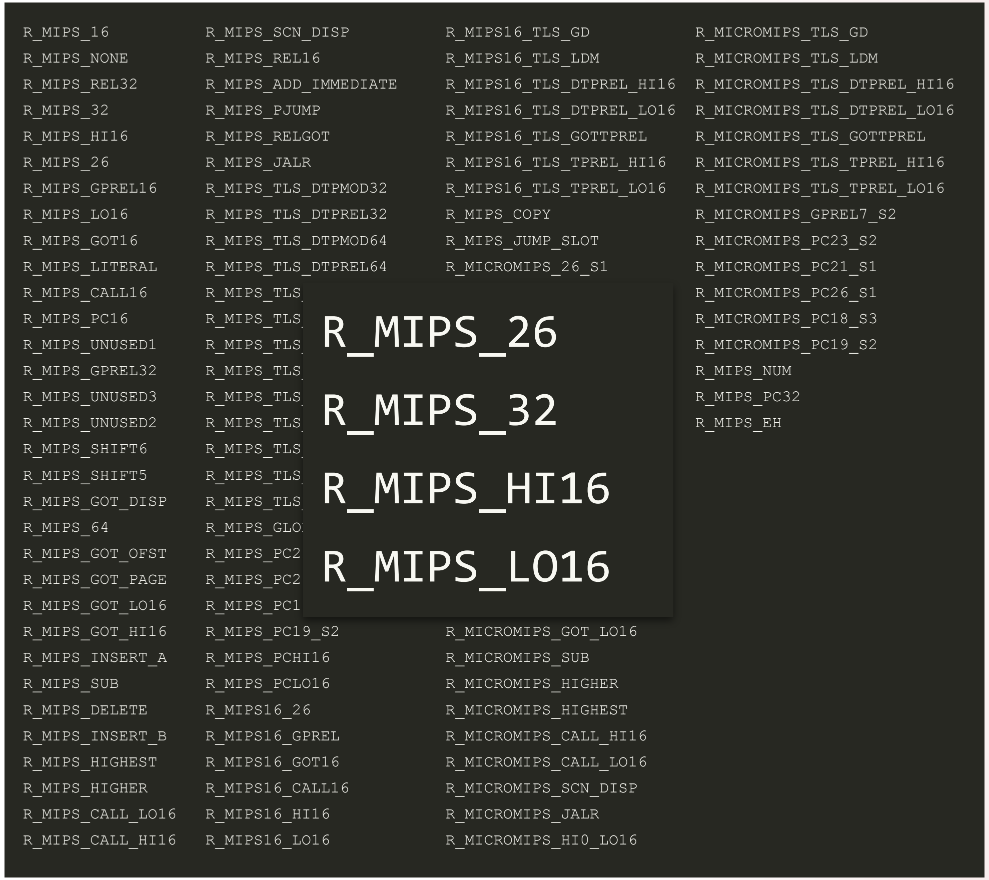
SHT_MIPS_REGINFO



https://github.com/97-things/97-things-every-programmer-should-know/tree/master/en/thing_53

Linkers

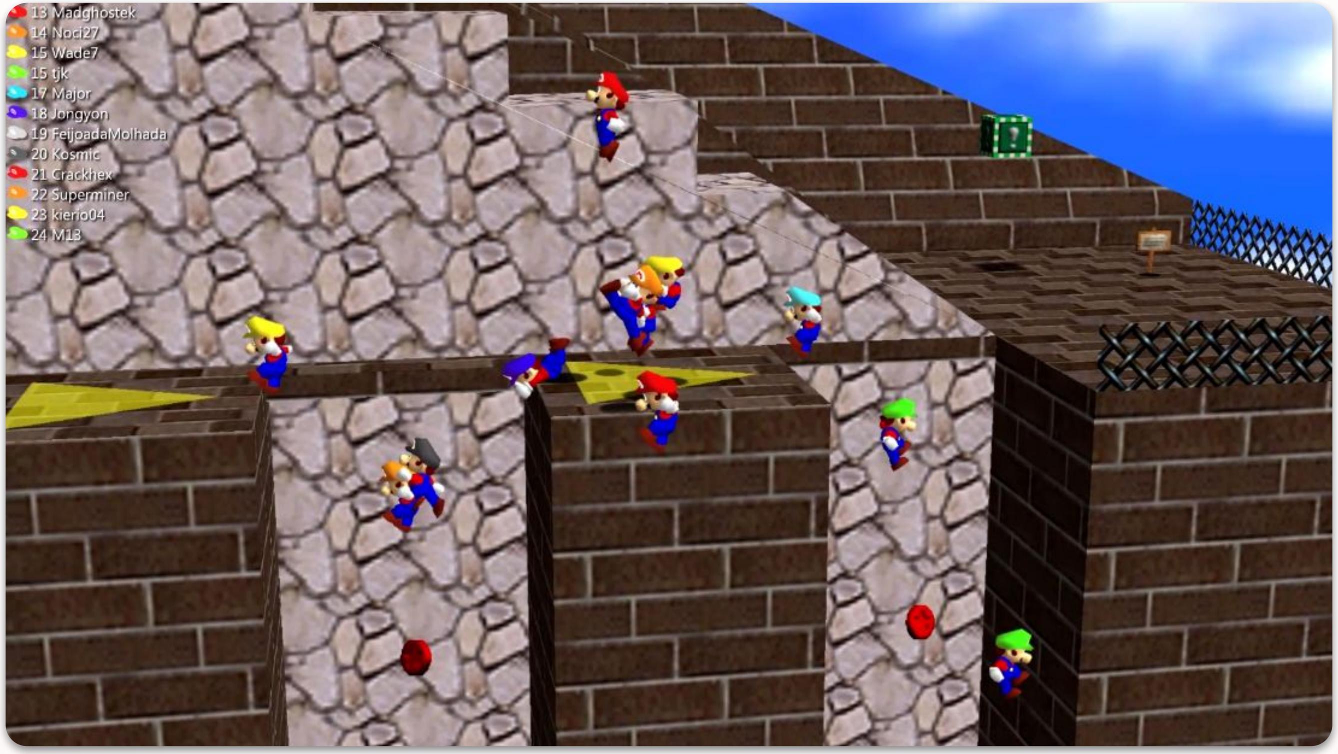
- Relocation table
- Don't need assembler
- ELF binary format



```
R_MIPS_16            R_MIPS_SCN_DISP          R_MIPS16_TLS_GD           R_MICROMIPS_TLS_GD
R_MIPS_NONE          R_MIPS_REL16             R_MIPS16_TLS_LDM          R_MICROMIPS_TLS_LDM
R_MIPS_REL32         R_MIPS_ADD_IMMEDIATE     R_MIPS16_TLS_DTPREL_HI16  R_MICROMIPS_TLS_DTPREL_HI16
R_MIPS_32            R_MIPS_PJUMP             R_MIPS16_TLS_DTPREL_LO16  R_MICROMIPS_TLS_DTPREL_LO16
R_MIPS_HI16          R_MIPS_RELGOT            R_MIPS16_TLS_GOTTPREL     R_MICROMIPS_TLS_GOTTPREL
R_MIPS_26            R_MIPS_JALR              R_MIPS16_TLS_TPREL_HI16   R_MICROMIPS_TLS_TPREL_HI16
R_MIPS_GPREL16       R_MIPS_TLS_DTPMOD32      R_MIPS16_TLS_TPREL_LO16   R_MICROMIPS_TLS_TPREL_LO16
R_MIPS_LO16          R_MIPS_TLS_DTPREL32      R_MIPS_COPY               R_MICROMIPS_GPREL7_S2
R_MIPS_GOT16         R_MIPS_TLS_DTPMOD64      R_MIPS_JUMP_SLOT          R_MICROMIPS_PC23_S2
R_MIPS_LITERAL       R_MIPS_TLS_DTPREL64      R_MICROMIPS_26_S1         R_MICROMIPS_PC21_S1
R_MIPS_CALL16        R_MIPS_TLS                                         R_MICROMIPS_PC26_S1
R_MIPS_PC16          R_MIPS_TLS                                         R_MICROMIPS_PC18_S3
R_MIPS_UNUSED1       R_MIPS_TLS                                         R_MICROMIPS_PC19_S2
R_MIPS_GPREL32       R_MIPS_TLS                                         R_MIPS_NUM
R_MIPS_UNUSED3       R_MIPS_TLS                                         R_MIPS_PC32
R_MIPS_UNUSED2       R_MIPS_TLS                                         R_MIPS_EH
R_MIPS_SHIFT6        R_MIPS_TLS
R_MIPS_SHIFT5        R_MIPS_TLS
R_MIPS_GOT_DISP      R_MIPS_TLS
R_MIPS_64            R_MIPS_GLO
R_MIPS_GOT_OFST      R_MIPS_PC2
R_MIPS_GOT_PAGE      R_MIPS_PC2
R_MIPS_GOT_LO16      R_MIPS_PC1
R_MIPS_GOT_HI16      R_MIPS_PC19_S2           R_MICROMIPS_GOT_LO16
R_MIPS_INSERT_A      R_MIPS_PCHI16            R_MICROMIPS_SUB
R_MIPS_SUB           R_MIPS_PCLO16            R_MICROMIPS_HIGHER
R_MIPS_DELETE        R_MIPS16_26              R_MICROMIPS_HIGHEST
R_MIPS_INSERT_B      R_MIPS16_GPREL           R_MICROMIPS_CALL_HI16
R_MIPS_HIGHEST       R_MIPS16_GOT16           R_MICROMIPS_CALL_LO16
R_MIPS_HIGHER        R_MIPS16_CALL16          R_MICROMIPS_SCN_DISP
R_MIPS_CALL_LO16     R_MIPS16_HI16            R_MICROMIPS_JALR
R_MIPS_CALL_HI16     R_MIPS16_LO16            R_MICROMIPS_HI0_LO16
```

```
R_MIPS_26
R_MIPS_32
R_MIPS_HI16
R_MIPS_LO16
```

Result



https://www.youtube.com/watch?v=yk9KrfLvgJQ

Result

- Very restricted feature set

- Still, benefits of using D over C:

syntax, reusable code, **`import("data.bin")`**

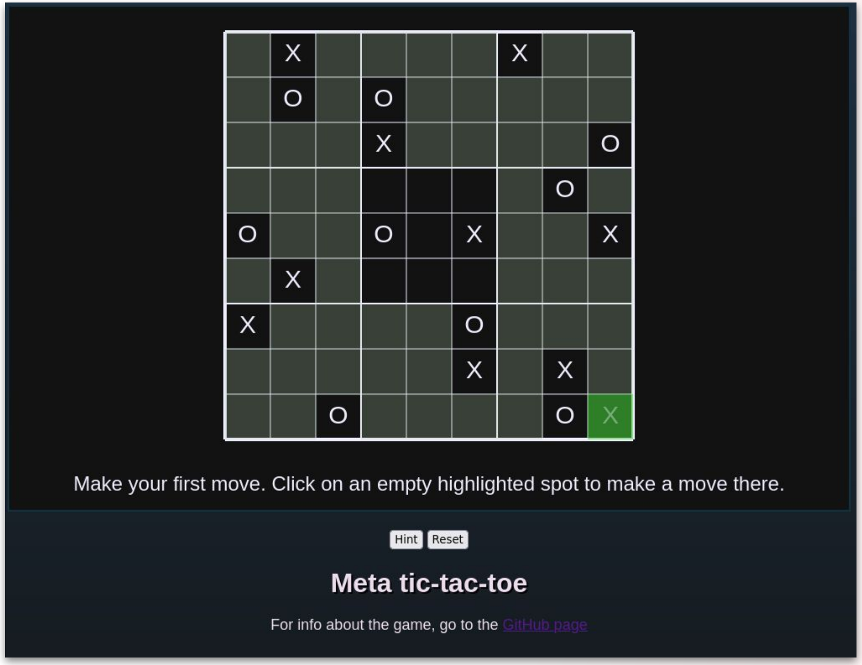
```
LLVM ERROR: Not supported instr:
<MCInst 0 <MCOperand Reg:30> <MCOperand Reg:25>>
```

D code

in the browser

D in the browser

- JavaScript
- Java Applet, Flash player
- Summer 2018: LDC adds WebAssembly
- `betterC` / Bare metal
- Ultimate tic-tac-toe game



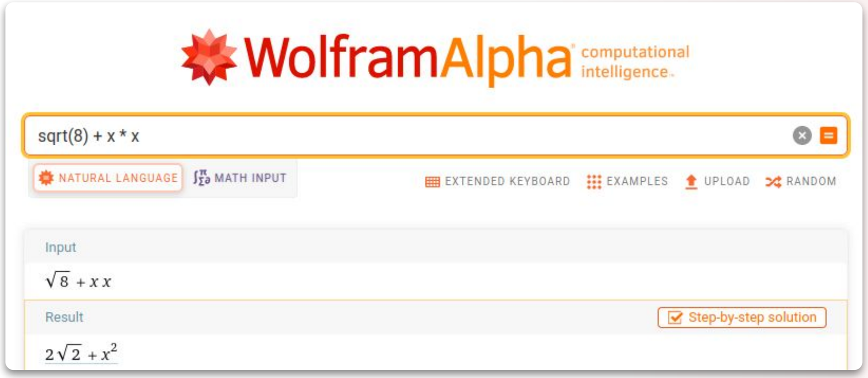
Make your first move. Click on an empty highlighted spot to make a move there.

Meta tic-tac-toe

For info about the game, go to the GitHub page

Exact math calculator

- Simplify formulas with variables
- Mathematica not open source
- Write my own



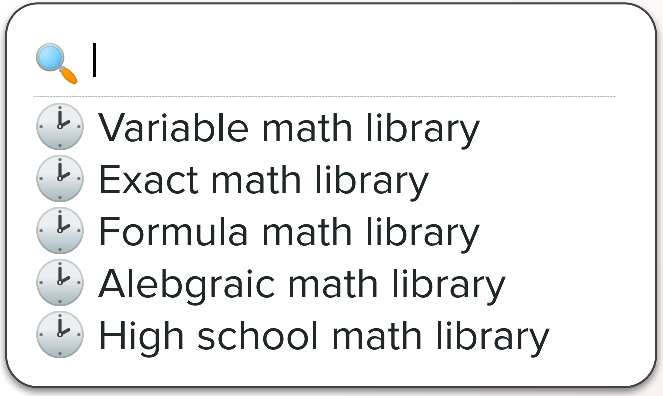
sqrt(8) + x * x

Input

$\sqrt{8} + x\,x$

Result

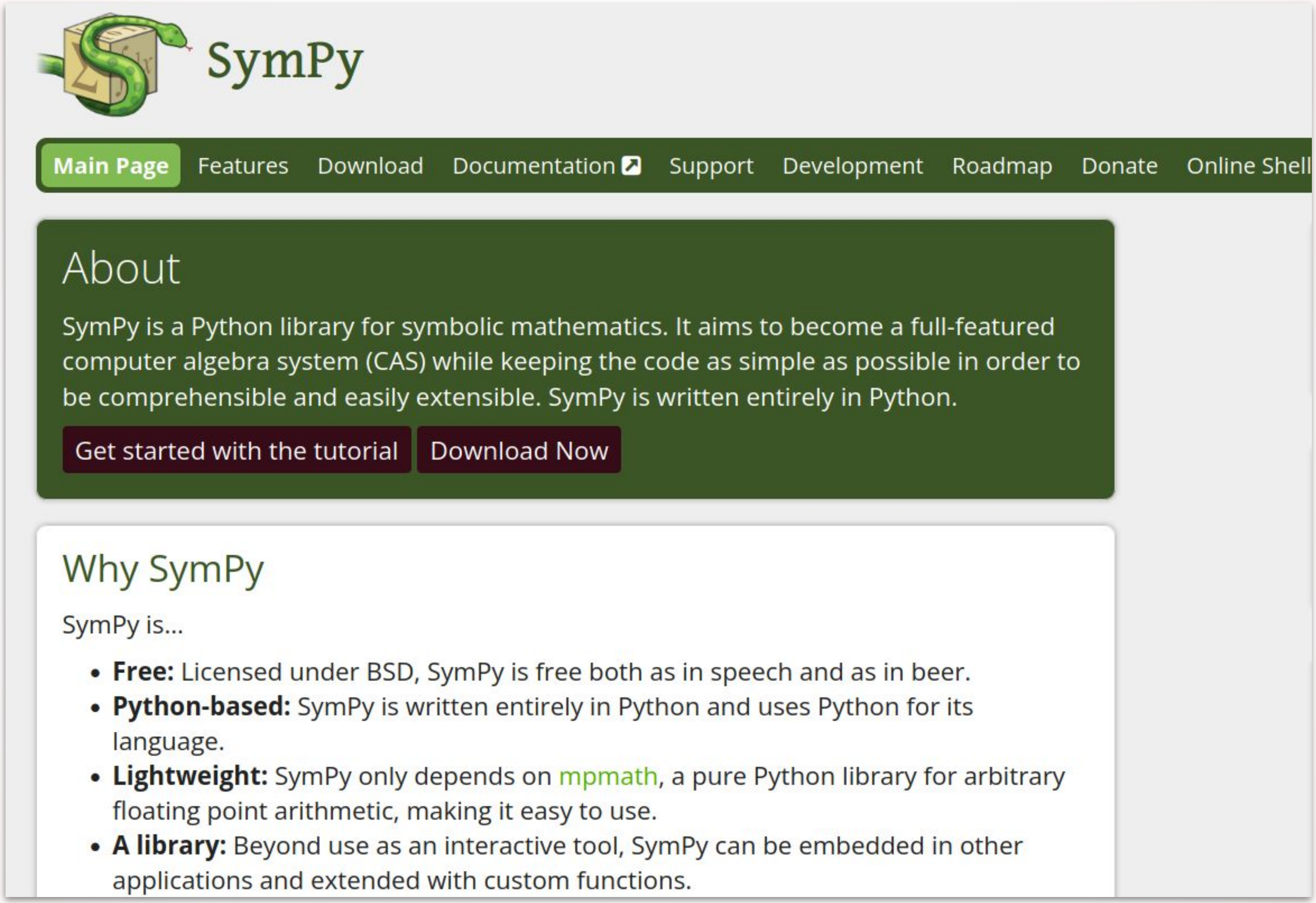
$2\sqrt{2} + x^2$



🔍 |

🕑 Variable math library
🕑 Exact math library
🕑 Formula math library
🕑 Alebgraic math library
🕑 High school math library

https://www.wolframalpha.com https://www.wolfram.com/mathematica/



About

SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible and easily extensible. SymPy is written entirely in Python.

Why SymPy

SymPy is...

- **Free:** Licensed under BSD, SymPy is free both as in speech and as in beer.
- **Python-based:** SymPy is written entirely in Python and uses Python for its language.
- **Lightweight:** SymPy only depends on mpmath, a pure Python library for arbitrary floating point arithmetic, making it easy to use.
- **A library:** Beyond use as an interactive tool, SymPy can be embedded in other applications and extended with custom functions.

https://www.sympy.org/en/index.html

Generic number types

- D has operator overloading
- Drop-in replacement for `float`

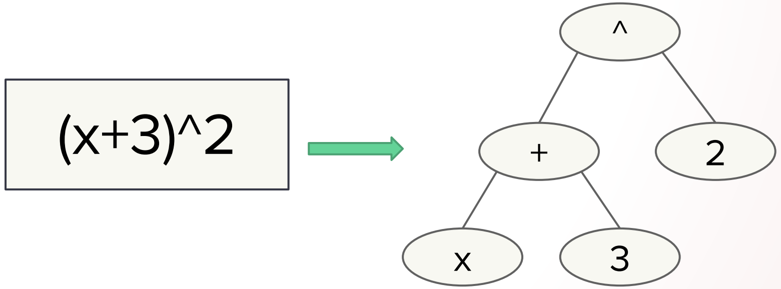
D

```d
T[2][2] matrixInverse2x2(T)(T[2][2] c)
{
    const T det = c[0][0] * c[1][1] - c[0][1] * c[1][0];
    const T invDet = 1 / det;
    T[2][2] res;
    res[0][0] = c[1][1] * invDet;
    res[0][1] = -c[0][1] * invDet;
    res[1][0] = -c[1][0] * invDet;
    res[1][1] = c[0][0] * invDet;
    return res;
}
```

Generic number types

- D has operator overloading
- Drop-in replacement for `float`
- Value type + tree data structure

```
                                                  D
void main()
{
    float x = 3;
    float y = x;
    y++;
    assert(x == 3);
}
```



Generic number types

- D has operator overloading
- Drop-in replacement for `float`
- Value type + tree data structure
- Inspired by `BigInt` and "Invariant strings"

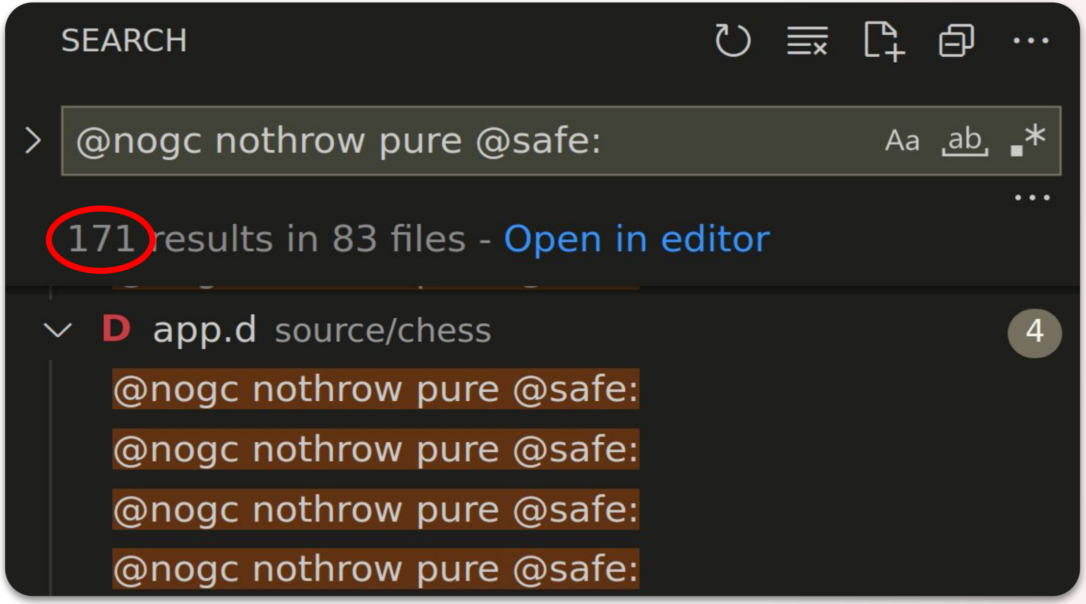
```d
struct MathNum
{
    int tag;
    immutable(MathNum)[] args;
}
```

D

https://digitalmars.com/articles/b01.html

WebAssembly

- I use function attributes a lot



```
@nogc nothrow pure @safe:
```

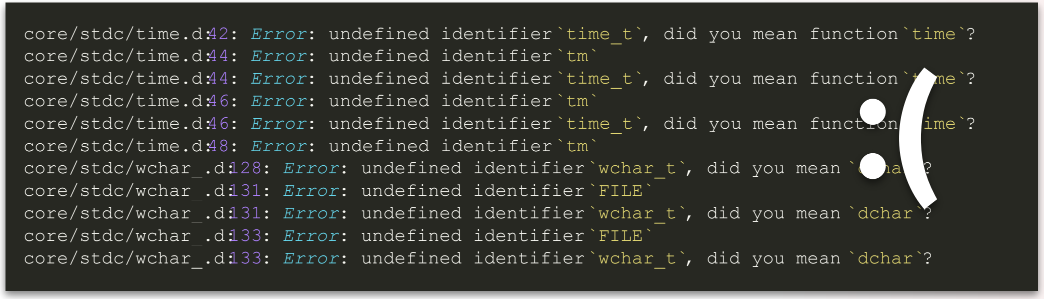
171 results in 83 files - Open in editor

app.d source/chess 4

```
@nogc nothrow pure @safe:
@nogc nothrow pure @safe:
@nogc nothrow pure @safe:
@nogc nothrow pure @safe:
```

WebAssembly

- I use function attributes a lot
- But this one uses GC
- Remove `-betterC` flag?
- druntime not ready



```
core/stdc/time.d:42: Error: undefined identifier `time_t`, did you mean function `time`?
core/stdc/time.d:44: Error: undefined identifier `tm`
core/stdc/time.d:44: Error: undefined identifier `time_t`, did you mean function `time`?
core/stdc/time.d:46: Error: undefined identifier `tm`
core/stdc/time.d:46: Error: undefined identifier `time_t`, did you mean function `time`?
core/stdc/time.d:48: Error: undefined identifier `tm`
core/stdc/wchar_.d:128: Error: undefined identifier `wchar_t`, did you mean `dchar`?
core/stdc/wchar_.d:131: Error: undefined identifier `FILE`
core/stdc/wchar_.d:131: Error: undefined identifier `wchar_t`, did you mean `dchar`?
core/stdc/wchar_.d:133: Error: undefined identifier `FILE`
core/stdc/wchar_.d:133: Error: undefined identifier `wchar_t`, did you mean `dchar`?
```

WebAssembly

- Solution: custom druntime
- GitHub: adamdruppe/webassembly
- Pass custom object.d to compiler

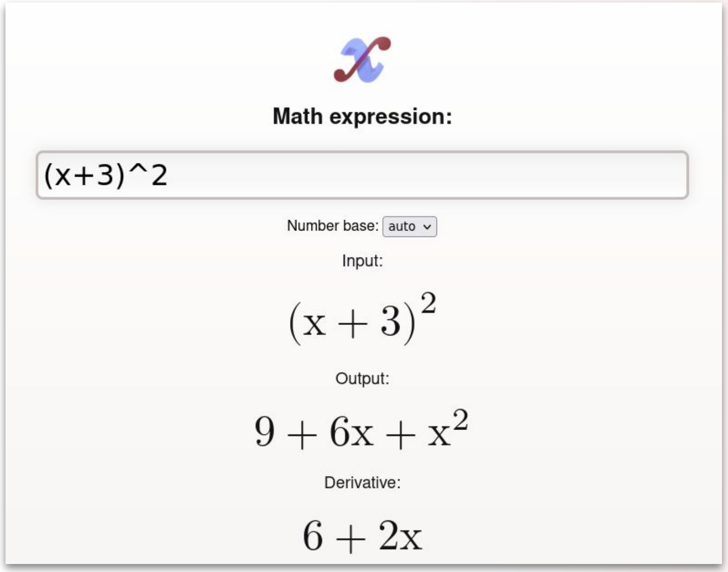
```d
module object;

alias string = immutable(char)[];
alias size_t = typeof(int.sizeof);
alias noreturn = typeof(*null);

extern(C) void _d_assert(string file, int line)
{
}
```

```
ldc2 -i -mtriple=wasm32-unknown-unknown-wasm -L-allow-undefined \
wasm/object.d app.d \
-of=app.wasm
```

https://github.com/adamdruppe/webassembly, https://github.com/hmmdyl/LWDR

Result

- Almost a programming language
- Leaks memory
- Limited implicit conversions


Math expression:

(x+3)^2

Number base: auto

Input:

$$(x+3)^2$$

Output:

$$9+6x+x^2$$

Derivative:

$$6+2x$$

```D
float[] array = [10, 20];
MathNum[] arrayB = [MathNum(10), MathNum(20)];
```

D code

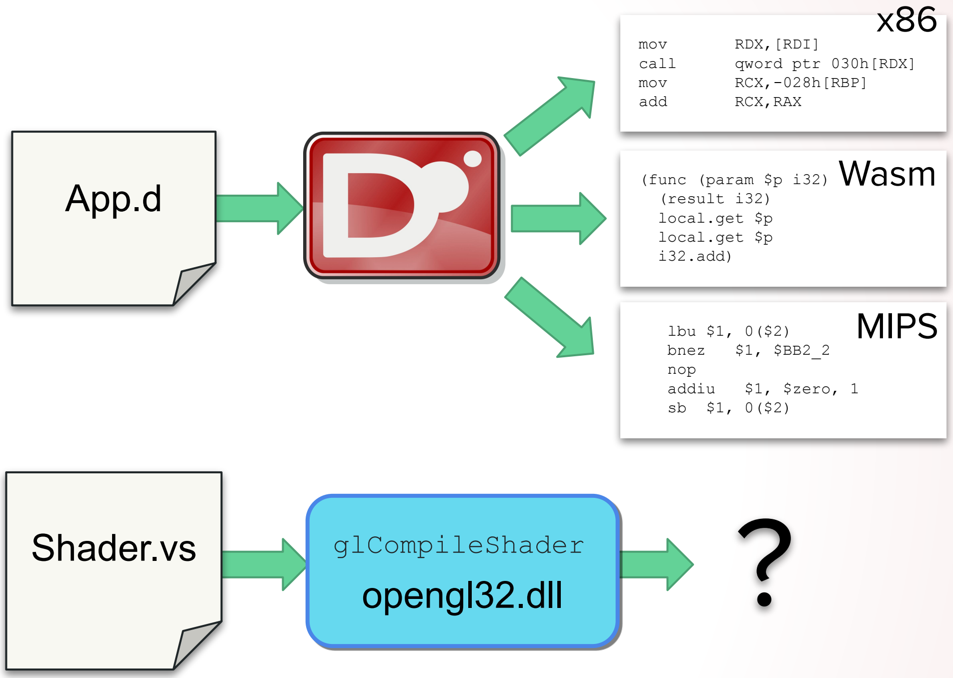
on the graphics card

(I didn't run D code on the graphics card)

Graphics cards

- GPUs have no defined ISA

- APIs: DirectX, OpenGL, Vulkan



```
mov         RDX,[RDI]
call        qword ptr 030h[RDX]
mov         RCX,-028h[RBP]
add         RCX,RAX
```

```
(func (param $p i32)
  (result i32)
   local.get $p
   local.get $p
   i32.add)
```

```
    lbu $1, 0($2)
    bnez   $1, $BB2_2
    nop
    addiu   $1, $zero, 1
    sb  $1, 0($2)
```

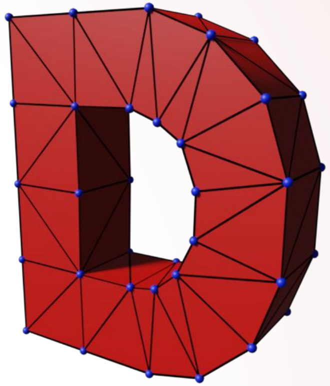
SPIR-V

Using OpenGL

- Inform GPU about struct layout

D

```d
struct Vertex
{
    float[3] position;
    ubyte[4] color;
}
```



GLSL

```glsl
#version 330
layout(location = 0) in vec3 position;
layout(location = 1) in vec4 color;

out vec3 fragColor;

uniform mat4 matrix;

void main() {
   gl_Position = matrix * vec4(position, 1.0);
   fragColor = color;
}
```

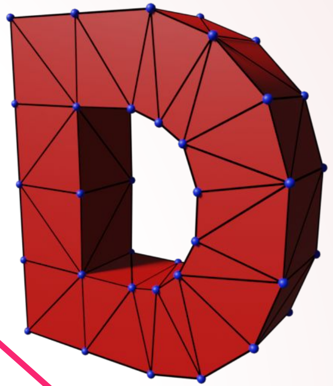
Using OpenGL

- Inform GPU about struct layout

- Very brittle

D
```
struct Vertex
{
    float[3] position;
    ubyte[4] color;
}
```

D
```
void setupVao()
{
    GLuint vao;
    glGenVertexArrays(1, &vao);
    glBindVertexArray(vao);

    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 16, 0);
    glEnableVertexAttribArray(1);
    glVertexAttribPointer(1, 4, GL_UNSIGNED_BYTE, GL_TRUE, 16, 12);
}
```

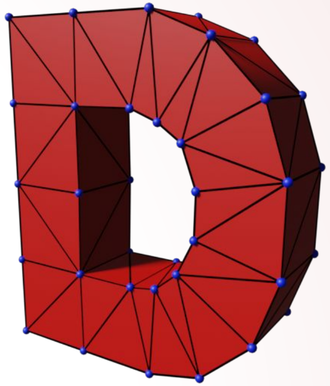


Oops!
Dennis Korpel

Metaprogramming

- Avoid magic numbers

```d
struct Vertex
{
    float[3] position;
    ubyte[4] color;
}
```



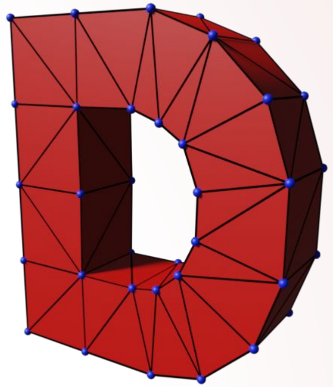
```d
glVertexAttribPointer(
    /*location*/ 0,
    /*components*/ Vertex.position.length,
    /*base type*/ GL_FLOAT,
    /*normalized*/ GL_FALSE,
    /*stride*/ Vertex.sizeof,
    cast(void*) Vertex.position.offsetof
);
```

Metaprogramming

- Avoid magic numbers

- Generate the right calls

```d
struct Vertex
{
    float[3] position;
    ubyte[4] color;
}
```



```d
import std.meta, std.traits;

void defineVaoAttributes(V)()
{
   static foreach(i; 0 .. V.tupleof.length)
   {{
       alias Arr = typeof(V.tupleof[i]);
       alias Elem = typeof(Arr.init[0]);
       glVertexAttribPointer(i, Arr.length, toGl!Elem, isIntegral!Elem,
           V.sizeof, cast(void*) V.tupleof[i].offsetof);
   }}
}
enum toGl(T) = [GL_UNSIGNED_BYTE, GL_FLOAT][staticIndexOf!(T, ubyte, float)];
```

Result

- Expand to support all types

- Automatically generate GLSL code

D

```d
void setupVao()
{
    GLuint vao;
    glGenVertexArrays(1, &vao);
    glBindVertexArray(vao);

    defineVaoAttributes!Vertex();
}
```

GLSL

```glsl
#version 330
layout(location = 0) in vec3 position;
layout(location = 1) in vec4 color;

out vec3 fragColor;

uniform mat4 matrix;

void main() {
   gl_Position = matrix * vec4(position, 1.0);
   fragColor = color;
}
```

Result

- Expand to support all types
- Automatically generate GLSL code
- Still some friction

D

```d
struct Buffer {
   bool b; /// 1 byte
   float[3][] array;
}
```

GLSL

```glsl
buffer Buffer {
  bool b; /// 4 bytes
  float[][3] array;
};
```

Wrapping up

- ❤️ to other programming languages

- Just one guy's perspective

- D is complex

- Reusable, robust, flexible code that runs everywhere

The Jack of all trades

Dennis Korpel - DConf 2022