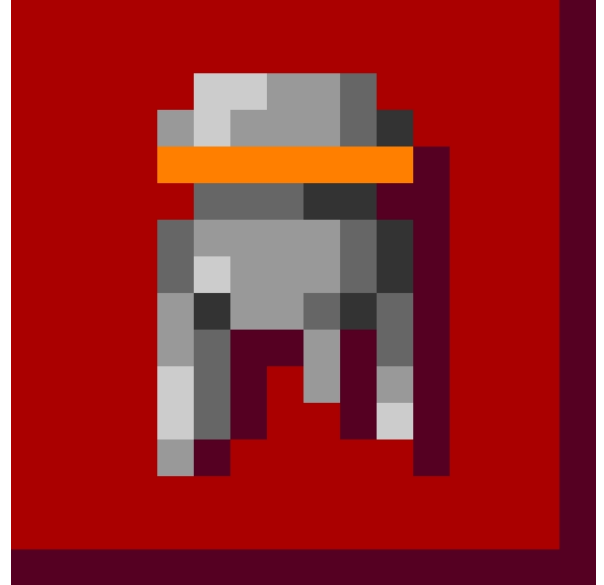


HAXE

Internship at Haxe

# About me

Aurel [ow - rel]



# Internship

4-year MEng degree

Required industrial placement

The logo for Imperial College London, featuring the text "Imperial College London" in white, bold, sans-serif font centered on a dark blue rectangular background.

**Imperial College  
London**

# The Haxe Foundation is Recruiting!

We are looking for people who are passionate about programming languages.

Article by [Nicolas Cannasse](#) on [2018-01-31](#).

[Comments](#)

The Haxe Foundation is Recruiting!

After a few years of working as a decentralized part-time team, I think that it is now a good time to start hiring some full-time people to work on Haxe. Thanks to our partners, we are able to hire for two full-time positions.

We are mostly looking for people who are passionate about programming languages and have a strong background in tech. No prior Haxe expertise is required as we can easily teach you Haxe. Of course, prior experience with Haxe would still be a plus.

## Compiler Developer



can I work remotely pls?

kind regards, Aurel

A close-up photograph of several small green seedlings with two leaves each, growing out of dark brown soil. The background is dark and out of focus.

# class HaxeFoundation extends Manpower

Powerful forces arrive from the East to support us

Article by [Simon Krajewski](#) on [2019-04-03](#).

[Comments](#)

## class HaxeFoundation extends Manpower

The Haxe Foundation is proud to announce that [Alexander Kuzmenko](#) will be working for us full-time starting right now! Alexander has been a member of the community for many years and has already made awesome contributions, like the [new PHP target](#) and the [null-safety](#) feature. We are excited to have his support on our quest towards Haxe 4 and beyond!

Furthermore, [Aurel Bily](#) is joining us as an intern for 6 months. He will be working on various projects, for instance an asynchronous sys-API. We are looking forward to the results of his internship!

## HF Internship

Topics of interest for a 6 month internship.

- Sys API : write unit tests and specification for current sys API, ensure specification is respected on all platforms
- Async Sys API : design, implement, document and test async Sys API on various platforms
- HashLink GC : look into improving performances of HashLink GC based on real world use cases (Northgard, Dead Cells)
- HashLink Profiling : allow HxScout to report both memory and cpu profiling information from HL JIT application.
- Ease the workflow to compile Hashlink to Android or iOS

# Intern work

noun (also **internship**)

**1** work that does not affect the productivity of the core team.

**2** (also **wasted time**)work “for the benefit of the community”.

ORIGIN early 16th cent. (as an adjective in the sense ‘internal’): from French **interne** (adjective), **interner** (verb), from Latin **internus** ‘inward, internal’. Current senses date from the 19th cent.



# Manual

Haxe 4 changes, part 1 #383

Edit

Haxe 4 changes, part 2 #387

Edit

Auto-generated define / metadata tables #395

Edit

Operators #402

Edit

Strings #406

Edit

Literals and constants #410

Edit

 **Merged** Simn merged 4 commits into `master` from `feature/literals-md`  on Jun 25

 Conversation 0

 Commits 4

 Checks 0

 Files changed 3

+107 -56 

```
- \begin{lstlisting}
- var myButton = new MySpecialButton();
- \end{lstlisting}
- We will explore type inference in detail later in [Type Inference](type-system-type-inference)
- that the variable myButton in the above code is known to be an instance of class MySpecialButton.
-
- The Haxe type system knows seven type groups:
-
- \begin{description}
- \item[\emph{Class instance}:] an object of a given class or interface
- \item[\emph{Enum instance}:] a value of a Haxe enumeration
- \item[\emph{Structure}:] an anonymous structure, i.e. a collection of named fields
- \item[\emph{Function}:] a compound type of several arguments and one return
- \item[\emph{Dynamic}:] a wildcard type which is compatible with any type
- \item[\emph{Abstract}:] a compile-time type which is represented by a different type at runtime
- \item[\emph{Monomorph}:] an unknown type which may later become a different type
- \end{description}
```

```
+ ```haxe
+ var myButton = new MySpecialButton(); // Haxe
+ ```
+ We will explore type inference in detail later in [Type Inference](type-system-type-inference)
+ that the variable myButton in the above code is known to be an instance of class MySpecialButton.
+
+ The Haxe type system knows seven type groups:
+
+ * Class instance: an object of a given class or interface
+ * Enum instance: a value of a Haxe enumeration
+ * Structure: an anonymous structure, i.e. a collection of named fields
+ * Function: a compound type of several arguments and one return
+ * Dynamic: a wildcard type which is compatible with any type
+ * Abstract: a compile-time type which is represented by a different type at runtime
+ * Monomorph: an unknown type which may later become a different type
+
+ We will describe each of these type groups and how they relate to each other in the next chapter
```

## LaTeXorcism #409

Edit

 Merged Aurel300 merged 16 commits into `master` from `feature/markdown` on Jun 24

 Conversation 0

 Commits 16

 Checks 0

 Files changed 518

+9,262 -20,748 

# Reformat std #8408

 Merged Simn merged 28 commits into `HaxeFoundation:development`

 Conversation 30

 Commits 28

 Checks 35



# Stdlib and Compiler

## Improvements

# Map.clear()


→ Haxe 4.0

```
var map = new Map<String, Int>();  
// add keys...  
map = []; // clear keys  
  
final map = new Map<String, Int>();  
// add keys...  
map = []; // error!
```

## Map.clear #8681

Edit

 **Merged** Aurel300 merged 18 commits into [HaxeFoundation:development](#) from [Aurel300:feature/map-clear](#)  on Aug 30

 Conversation 3

 Commits 18

 Checks 44

 Files changed 53

+416 -2 

# UTC-based Date methods

→ Haxe 4.0

```
date.getUTCHours();  
date.getUTCMinutes();  
date.getUTCSeconds();  
date.getUTCFullYear();  
date.getUTCMonth();  
date.getUTCDate();  
date.getUTCDay();  
date.getTimezoneOffset();
```

## Date improvements #8508

Edit

 Merged Simn merged 29 commits into [HaxeFoundation:development](#) from [Aurel300:issue/7303](#)  on Jul 10

 Conversation 8

 Commits 29

 Checks 35

 Files changed 17

+719 -162 



```

- type strict_defined =
-   | AbsolutePath
-   | AdvancedTelemetry
-   | AnnotateSource
-   (* | Analyzer *)
-   | As3
-   | CheckXmlProxy
-   | CoreApi
-   | CoreApiSerialize
-   | Cppia
-   | NoCppiaAst
-   | Dce
-   | DceDebug
-   | Debug
-   | DisableUnicodeStrings
- let infos = function
-   | AbsolutePath -> "absolute_path",("Print absolute file path
-   | AdvancedTelemetry -> "advanced-telemetry",("Allow the SWF t
-   | AnnotateSource -> "annotate_source",("Add additional commen
-   (* | Analyzer -> "analyzer",("Use static analyzer for optimiz
-   | As3 -> "as3",("Defined when outputting flash9 as3 source co
-   | CheckXmlProxy -> "check_xml_proxy",("Check the used fields
-   | CoreApi -> "core api",("Defined in the core api context".[]
+ [
+   {
+     "name": "AbsolutePath",
+     "define": "absolute_path",
+     "doc": "Print absolute file path in trace output"
+   },
+   {
+     "name": "AdvancedTelemetry",
+     "define": "advanced-telemetry",
+     "doc": "Allow the SWF to be measured with Monocle tool",
+     "platforms": ["flash"]
+   },
+   {
+     "name": "AnnotateSource",
+     "define": "annotate_source",
+     "doc": "Add additional comments to generated source code",
+     "platforms": ["cpp"]
+   },
+   {

```

# Auto-generate defines and metas from JSON files

## #8195

[Edit](#)
 Merged

 Simn merged 19 commits into [HaxeFoundation:development](#) from [Aurel300:feature/autogen-define-meta](#)  on May 23

```
0 example . haxe --server-listen 1234 &  
[1] 1014  
0 example . haxe --main Main.hx --js out.js --connect 1234  
0 example . █
```



# hxb file format

---

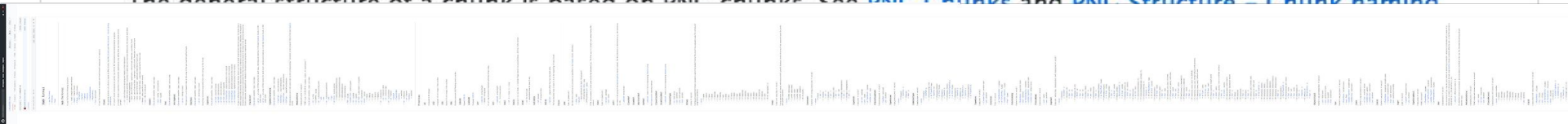
A `hxb` file has the following structure:

- : `"hxb1"` (magic bytes and version)
- **Header** : header chunk
- **Chunk...** : data chunks:
  - **StringPool**
  - **DocPool**
  - **TypeList**
  - **FieldList**
  - **TypeDeclarations**
  - **ModuleExtra**
- **End** : end chunk

A `hxb` file corresponds to exactly one Haxe module ( `module_def` in `type.ml` ).

## Chunk

The general structure of a chunk is based on PNG chunks. See [PNG Chunks](#) and [PNG Structure – Chunk naming](#)



```

Foobar.hx
Foobar.hx
1 class Foobar {
2   public static func
3   for (i in 0...3)
4     trace('foo $i $
5 }
6
7
8 class Boofar {
9   public static func
10  return "nice";
11 }
12
13
Line: 4:30 Haxe Sof

```

Foobar.hxb
0 bytes out of 1.7 kilobytes

0	68 78 62 31 73 00 00 00 48 48 44 52 01 00 06 46 6F 6F 62 61 72 29 2F 44 65 76 50 72 6F 6A 65 63 74 73 2F 53 74 75 66 66	hxbis HHDR Foobar)/DevProjects/Stuff
40	2F 68 61 78 65 2F 31 37 35 20 68 78 62 2F 46 6F 62 61 72 2E 68 78 AA 85 40 67 16 09 03 A6 F1 4E CB 28 8D AD 6A 88 00	/haxe/175-hxb/Foobar.hx..@g...N(..j
80	00 00 00 C0 E8 AC 5C D7 41 00 00 01 00 00 02 14 75 6E 73 61 66 65 5F 73 74 72 69 6E 67 5F 63 6F 6E 63 61 74 01 08 68 61	...\.A unsafe_string_concat ha
120	73 5F 61 6E 6F 6E 01 67 45 23 01 A7 00 00 00 53 54 52 49 1A 05 3A 75 73 65 64 09 46 6F 6F 62 61 72 2E 68 78 05 3A 68 65	s_anon gE# . STRI :used Foobar.hx :ke
160	65 70 05 74 72 61 63 65 01 76 05 69 6E 66 6F 73 04 66 6F 6F 28 01 20 02 6F 68 08 66 69 6C 65 4E 61 6D 65 01 3F 0A 6C 69	ep trace v infos foo ok fileName ? li
200	6E 65 4E 75 6D 62 65 72 09 63 6C 61 73 73 4E 61 6D 65 06 46 6F 6F 62 61 72 0A 6D 65 74 68 6F 64 4E 61 6D 65 04 6D 61 69	LineNumber className Foobar methodName mai
240	6E 00 3A 64 69 72 65 63 74 6C 79 55 73 65 64 04 6E 69 63 65 04 68 61 78 65 03 4C 6F 67 66 53 74 72 69 6E 67 04 56 6F 69	:directlyUsed nice haxe Log String Voi
280	64 04 4E 75 6C 6C 03 49 6E 74 08 50 6F 73 49 6E 66 6F 73 06 42 6F 6F 66 61 72 67 45 23 01 01 00 00 00 64 4F 43 53 00 67	d Null Int PosInfos BoofargE# dOCS g
320	45 23 01 27 00 00 00 54 59 58 4C 02 01 12 13 00 14 00 03 00 15 00 16 00 17 01 01 12 18 02 00 00 DF 01 01 06 18 00 19 EB	E# ' TYPL
360	00 F6 02 06 FA 01 00 00 00 67 45 23 01 14 00 00 00 46 4C 44 4C 01 0F 10 DB 01 01 17 D6 00 00 01 08 DA 00 F2 02 17 B4	. . gE# FLDL . . . .
400	02 00 67 45 23 01 0E 00 00 00 78 54 52 01 02 2E 05 00 01 0C 7F EF 00 6C 02 04 7E 00 67 45 23 01 D3 04 00 00 54 59 50 45	gE# xTRA .. . ~gE# . TYPE
440	00 01 00 00 00 DF 01 01 00 00 00 00 00 00 0C 7F 00 01 02 00 10 DA 01 00 00 01 15 00 00 0C 7F 18 03 12 05 08 7F 09 04	. . . .
480	7F 3B A0 01 7F 03 08 02 04 00 0A 05 01 0D 7E 01 06 7F 0C 7F 0A A0 01 02 03 00 03 00 03 00 02 06 04 7E 07 AC 01 00 00	;
520	00 0C 7D 05 00 01 04 7E 7B 08 01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01 12 05 0B 01 09 04 01 09 C2 01 01 08 08 00 04 7E	. . ~{. . . ~Z. . . .
560	00 C8 01 00 04 7E 00 CC 01 04 7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02 01 04 7E CC 00 A1 01 01 0B B4 7F 7F 0A 00 00	. . ~w. . . .
600	04 0C 7D CC 00 A1 01 01 0C B4 7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01 0E B4 7F 7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01	. . . .
640	09 02 00 A0 01 0C 7F 00 02 06 04 7E 12 05 0B 7F 09 04 7F 0A 01 7F 03 08 02 04 00 0A 05 01 0D 7E 01 06 7F 0C 7F 0A 01 02	. . . .
680	03 00 03 00 03 00 02 06 04 7E 07 AC 01 00 00 01 0C 7D 05 0B 01 04 7E 7B 00 01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01	. . . .
720	12 05 0B 01 09 04 01 09 C2 01 01 08 08 00 04 7E 00 C8 01 00 04 7E 00 CC 01 04 7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02	. . . .
760	01 04 7E CC 00 A1 01 01 0B B4 7F 7F 0A 00 00 04 0C 7D CC 00 A1 01 01 0C B4 7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01	~w. . . .
800	0E B4 7F 7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01 09 02 00 A0 01 0C 7F 00 0A 01 12 05 0B 7F 09 04 7F 00 A0 01 7F 03 08 02	. . . .
840	04 00 0A 05 01 0D 7E 01 06 7F 0C 7F 0A 01 02 03 00 03 00 03 00 02 06 04 7E 07 AC 01 00 00 02 0C 7D 05 0B 01 04 7E	. . . .
880	7B 00 01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01 12 05 0B 01 09 04 01 09 C2 01 01 08 08 00 04 7E 00 C8 01 00 04 7E 00 CC	{. . . ~Z. . . .
920	01 04 7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02 01 04 7E CC 00 A1 01 01 0B B4 7F 7F 0A 00 00 04 0C 7D CC 00 A1 01 01	~w. . . .
960	0C B4 7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01 0E B4 7F 7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01 09 02 00 A0 01 0C 7F 00	. . . .
1000	A0 01 0C 7F 6C D2 01 08 00 0C 7F 59 DA 01 01 00 0C 7F 18 01 18 03 18 01 12 05 0B 7F 09 04 7F 3B A0 01 7F 03 08 02 04 00	. l. y. . . ;
1040	0A 05 01 0D 7E 01 06 7F 0C 7F 00 A0 01 02 03 00 03 00 00 02 06 04 7E 07 AC 01 00 00 00 0C 7D 05 0B 01 04 7E 7B 00	. . . .
1080	01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01 12 05 0B 01 09 04 01 09 C2 01 01 08 08 00 04 7E 00 C8 01 00 04 7E 00 CC 01 04	. . . ~Z. . . .
1120	7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02 01 04 7E CC 00 A1 01 01 0B B4 7F 7F 0A 00 00 04 0C 7D CC 00 A1 01 01 0C B4	~w. . . .
1160	7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01 0E B4 7F 7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01 09 02 00 A0 01 0C 7F 00 A0 01	. . . .
1200	0C 7F 00 D2 01 18 01 12 05 0B 7F 09 04 7F 0A 00 01 7F 03 08 02 04 00 0A 05 01 0D 7E 01 06 7F 0C 7F 00 A0 01 02 03 00 03	. . . .
1240	00 03 00 00 02 06 04 7E 07 AC 01 00 00 01 0C 7D 05 0B 01 04 7E 7B 00 01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01 12 05 0B	. . . .
1280	01 09 04 01 09 C2 01 01 08 08 00 04 7E 00 C8 01 00 04 7E 00 CC 01 04 7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02 01 04 7E	. . . ~w. . . .
1320	CC 00 A1 01 01 0B B4 7F 7F 0A 00 00 02 00 04 0C 7D CC 00 A1 01 01 0C B4 7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01 0E B4 7F	. . . .
1360	7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01 09 02 00 A0 01 0C 7F 00 A0 01 0C 7F 00 D2 01 18 01 12 05 0B 7F 09 04 7F 00 A0 01	. . . .
1400	7F 03 08 02 04 00 0A 05 01 0D 7E 01 06 7F 0C 7F 0A 01 02 03 00 03 00 00 02 06 04 7E 07 AC 01 00 00 02 0C 7D 05 0B	. . . .
1440	B0 01 04 7E 7B 00 01 00 02 07 04 7E 06 B2 01 04 7E 7A B2 01 12 05 0B 01 09 04 01 09 C2 01 01 08 08 00 04 7E 00 C8 01 00	. ~{. . . ~Z. . . .
1480	04 7E 00 CC 01 04 7E 77 CC 01 10 04 09 AD 7F 7F 0A 00 00 02 01 04 7E CC 00 A1 01 01 0B B4 7F 7F 0A 00 00 04 0C 7D CC	. . ~w. . . .
1520	00 A1 01 01 0C B4 7F 7F 0A 00 00 02 00 04 7E CC 00 A1 01 01 0E B4 7F 7F 0A 00 00 02 0F 04 7E CC 00 A1 01 01 09 02 00 A0	. . . .
1560	01 0C 7F 00 A0 01 0C 7F 00 D2 01 0C 7F 6C D2 01 0C 7F 7C DA 01 00 01 00 00 00 00 02 10 00 3E F6 02 00 00 00 F6 02	. . . l.  . . . y. .
1600	00 00 00 00 00 00 00 04 7E 00 00 00 01 15 00 04 7E 18 01 23 00 02 11 04 7E 18 01 23 00 02 11 04 7E 2E EA 02 0A 79 EA 02	. . . # ~>. y. .
1640	04 7E 59 F2 02 01 00 04 7E 18 01 23 00 02 11 04 7E 2E EA 02 0A 79 EA 02 0A 7C F2 02 00 0A 79 EA 00 00 67 45 23 01 00	~Y. . # ~.. y.  . . gE#
1680	00 00 00 48 45 4E 44 67 45 23 01	HENDgE#

# hxb

→ Haxe 4.1 ?

Binary format for serialising modules, AST, and typed AST

Specification:

[github.com/Aurel300/hxb](https://github.com/Aurel300/hxb)

WIP compiler branch:

[github.com/HaxeFoundation/haxe/tree/hxb](https://github.com/HaxeFoundation/haxe/tree/hxb)

asys

asynchronous system APIs

# Current system APIs

No asynchrony (have to create threads manually)

Lacking file system functionality (e.g. chmod)

Hard-to-use TCP/UDP sockets

Hard-to-use processes

No IPC sockets

# Solutions?

How to design the API?

How to implement it consistently across many different sys targets?

How to represent asynchrony?



# fs.rename(oldPath, newPath, callback) [src] #

## ► History

- `oldPath` `<string>` | `<Buffer>` | `<URL>`
- `newPath` `<string>` | `<Buffer>` | `<URL>`
- `callback` `<Function>`
  - `err` `<Error>`

Asynchronously rename file at `oldPath` to the pathname provided as `newPath`. In the case that `newPath` already exists, it will be overwritten. If there is a directory at `newPath`, an error will be raised instead. No arguments other than a possible exception are given to the completion callback.

See also: `rename(2)`.

```
fs.rename('oldFile.txt', 'newFile.txt', (err) => {  
  if (err) throw err;  
})
```





libuv

```
int uv_fs_rename(uv_loop_t* loop, uv_fs_t* req, const char* path, const  
char* new_path, uv_fs_cb cb)
```

Equivalent to `rename(2)`.



HAXE

eval

```
static function rename(oldPath:FilePath, newPath:FilePath):Void;
```

```
let rename = vfun2 (fun oldPath newPath →  
  ▶ let oldPath = decode_string oldPath in  
  ▶ let newPath = decode_string newPath in  
  ▶ wrap_sync (Uv.fs_rename_sync (loop ()) oldPath newPath);  
  ▶ vnull  
)
```

```
external fs_rename :: t_loop -> string -> string -> unit_cb -> unit uv_result = "w_fs_rename"
```

```
FS_WRAP2(rename, String_val, String_val, handle_fs_cb);
```

```
#define FS_WRAP(name, sign, locals, precall, call, handler) \-  
  > CAMLprim value w_fs_##name(value loop, sign, value cb) { \-  
  > > CAMLparam2(loop, cb); \-  
  > > locals; \-  
  > > UV_ALLOC_REQ(req, uv_fs_t, cb); \-  
  > > precall \-  
  > > UV_ERROR_CHECK_C(uv_fs_##name(Loop_val(loop), Fs_val(req), call, handler), UV_FREE_REQ(Fs_val(req))); \-  
  > > UV_SUCCESS_UNIT; \-  
  > } \-  
  > CAMLprim value w_fs_##name##_sync(value loop, sign) { \-  
  > > CAMLparam1(loop); \-  
  > > locals; \-  
  > > UV_ALLOC_CHECK(req, uv_fs_t); \-  
  > > precall \-  
  > > UV_ERROR_CHECK_C(uv_fs_##name(Loop_val(loop), Fs_val(req), call, NULL), free(Fs_val(req))); \-  
  > > UV_ERROR_CHECK_C(Fs_val(req)→result, { uv_fs_req_cleanup(Fs_val(req)); free(Fs_val(req)); }); \-  
  > > CAMLlocal1(ret); \-  
  > > ret = handler ##_sync(Fs_val(req)); \-  
  > > uv_fs_req_cleanup(Fs_val(req)); \-  
  > > free(Fs_val(req)); \-  
  > > UV_SUCCESS(ret); \-  
  > } \-
```

And similarly on HL and Neko



# demo

Asynchronous file reading

# demo

Asynchronous DNS and IP address manipulation

# demo

TCP echo server

# demo

Process communication with ffmpeg

# demo

UDP video stream

# asys

File system operations

TCP/UDP/IPC sockets

DNS

Processes

Message passing (including open sockets) across Haxe processes

soon™

asys\*

→ Haxe 4.1 ?

Haxe API: feedback welcome! (PR #8832)

Targets mostly finished™:

HashLink (#306)

Eval (#8831)

Neko (#201)

\* better name?



# Side project

ammer

# \$250 – Common HL/HXCPP native C API binding format #2



larsiusprime opened this issue on Oct 3, 2018 · 29 comments



larsiusprime commented on Oct 3, 2018 • edited ▾

Owner + 😊 ...

Introductory Information:

[Hashlink](#) is basically the successor [Neko](#) in terms of Haxe VM targets, with the added advantage of having both a compiled-C and a bytecode target.

[HXCPP](#) is the traditional workhorse of Haxe native C++ targets.

Hashlink promises some advantages to workflow -- particularly faster compilation speed during development, but it's still relatively new and most of the C/C++ bindings for existing libraries for Haxe native targets are still written for HXCPP, and HXCPP is a solid well-maintained target that will surely see continued use alongside HashLink. This puts native extension library maintainers in a dilemma, do we support HXCPP or Hashlink, or both? Right now supporting both means maintaining two parallel sets of bindings, which is not only double the work, but also fertile ground for subtle duplication/oops-I-forgot-to-update-the-other-one errors.

This bounty is for developing a *common format* for writing native extensions to C API's for Haxe, that is compatible with both HXCPP and HashLink.

This is a separate issue from *automatically* generating native extensions with minimal effort from, say, raw C files. It's okay for the sake of this bounty if the accepted solution requires a bit of manual labor in writing the bindings themselves, so long as each binding doesn't have to be written *twice*, once for HXCPP and once for HashLink.

```

//-----PIPES-----
-
CAMLprim value w_pipe_init(value loop, value ipc) {-
  CAMLparam2(loop, ipc);-
  UV_ALLOC_CHECK(handle, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(handle)), - alloc_data());-
  UV_ERROR(0);-
  UV_SUCCESS(handle);-
}-
-
CAMLprim value w_pipe_open(value pipe, value fd) {-
  CAMLparam2(pipe, fd);-
  UV_ERROR_CHECK(uv_pipe_open(Pipe_val(pipe), Int_val(fd)), - alloc_data());-
  UV_SUCCESS_UNIT;-
}-
-
CAMLprim value w_pipe_accept(value loop, value server) {-
  CAMLparam2(loop, server);-
  UV_ALLOC_CHECK(client, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(server)), - alloc_data());-
  UV_ERROR(0);-
  UV_ERROR_CHECK_C(uv_accept(Stream_val(server), Stream_val(client)), - alloc_data());-
  UV_SUCCESS(client);-
}-
-
CAMLprim value w_pipe_bind_ipc(value handle, value path) {-
  CAMLparam2(handle, path);-
  UV_ERROR_CHECK(uv_pipe_bind(Pipe_val(handle), String_val(path)), - alloc_data());-
  UV_SUCCESS_UNIT;-
}-
-
CAMLprim value w_pipe_connect_ipc(value handle, value req, value connect_t, cb) {-
  CAMLparam3(handle, path, cb);-
  UV_ALLOC_REQ(req, uv_connect_t, cb);-
  UV_ERROR_CHECK(uv_pipe_connect(Connect_val(req), Pipe_val(handle), req, connect_t, cb), - alloc_data());-
  UV_SUCCESS(connect_t);-
}-

```

```

//-----PIPES-----
-
static value w_pipe_init(value loop, value ipc) {-
  UV_ALLOC_CHECK(handle, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(ipc)), - alloc_data());-
  UV_HANDLE_DATA(Pipe_val(handle)) = alloc_data();-
  if (UV_HANDLE_DATA(Pipe_val(handle)) == NULL)-
    UV_ERROR(0);-
  return handle;-
}-
DEFINE_PRIM(w_pipe_init, 2);-
-
static value w_pipe_open(value pipe, value fd) {-
  UV_ERROR_CHECK(uv_pipe_open(Pipe_val(pipe), Int_val(fd)), - alloc_data());-
  return val_null;-
}-
DEFINE_PRIM(w_pipe_open, 2);-
-
static value w_pipe_accept(value loop, value server) {-
  UV_ALLOC_CHECK(client, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(server)), - alloc_data());-
  UV_HANDLE_DATA(Pipe_val(client)) = alloc_data();-
  if (UV_HANDLE_DATA(Pipe_val(client)) == NULL)-
    UV_ERROR(0);-
  UV_ERROR_CHECK_C(uv_accept(Stream_val(server), Stream_val(client)), - alloc_data());-
  return client;-
}-
DEFINE_PRIM(w_pipe_accept, 2);-
-
static value w_pipe_bind_ipc(value handle, value path) {-
  UV_ERROR_CHECK(uv_pipe_bind(Pipe_val(handle), String_val(path)), - alloc_data());-
  return val_null;-
}-
DEFINE_PRIM(w_pipe_bind_ipc, 2);-
-
static value w_pipe_connect_ipc(value handle, value req, value connect_t, cb) {-
  UV_ALLOC_REQ(req, uv_connect_t, cb);-
  UV_ERROR_CHECK(uv_pipe_connect(Connect_val(req), Pipe_val(handle), req, connect_t, cb), - alloc_data());-
  return connect_t;-
}-

```

```

//-----PIPES-----
-
HL_PRIM uv_pipe_t **HL_NAME(w_pipe_init)(uv_loop_t **loop, uv_ipc_t **ipc) {-
  UV_ALLOC_CHECK(handle, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(ipc)), - alloc_data());-
  if ((UV_HANDLE_DATA(Pipe_val(handle)) == alloc_data())-
    UV_ERROR(0);-
  return handle;-
}-
DEFINE_PRIM(_PIPE, w_pipe_init, _LOOP_BOOL);-
-
HL_PRIM void HL_NAME(w_pipe_open)(uv_pipe_t **pipe, int fd) {-
  UV_ERROR_CHECK(uv_pipe_open(Pipe_val(pipe), fd));-
}-
DEFINE_PRIM(_VOID, w_pipe_open, _PIPE_I32);-
-
HL_PRIM uv_pipe_t **HL_NAME(w_pipe_accept)(uv_loop_t **loop, uv_server_t **server) {-
  UV_ALLOC_CHECK(client, uv_pipe_t);-
  UV_ERROR_CHECK_C(uv_pipe_init(Loop_val(loop), Pipe_val(server)), - alloc_data());-
  if ((UV_HANDLE_DATA(Pipe_val(client)) == alloc_data())-
    UV_ERROR(0);-
  UV_ERROR_CHECK_C(uv_accept(Stream_val(server), Stream_val(client)), - alloc_data());-
  return client;-
}-
DEFINE_PRIM(_PIPE, w_pipe_accept, _LOOP_PIPE);-
-
HL_PRIM void HL_NAME(w_pipe_bind_ipc)(uv_pipe_t **handle, uv_path_t **path) {-
  UV_ERROR_CHECK(uv_pipe_bind(Pipe_val(handle), path));-
}-
DEFINE_PRIM(_VOID, w_pipe_bind_ipc, _PIPE_BYTES);-
-
HL_PRIM void HL_NAME(w_pipe_connect_ipc)(uv_pipe_t **handle, uv_req_t **req, uv_connect_t **connect_t, uv_cb_t **cb) {-
  UV_ALLOC_REQ(req, uv_connect_t, cb);-
  uv_pipe_connect(Connect_val(req), Pipe_val(handle), req, connect_t, cb);-
}-
DEFINE_PRIM(_VOID, w_pipe_connect_ipc, _PIPE_BYTES_CB);-

```



```

function·new(ipc:Bool);-
function·open(fd:Int):Void;-
function·connectIpc(path:String,·cb:Callback<NoData>):Void;-
function·bindIpc(path:String):Void;-
function·accept():Pipe;-
function·writeHandle(data:Bytes,·handle:eval.uv.Stream,·cb:Callback<NoData>):V
function·pendingCount():Int;-
function·acceptPending():PipeAccept;-
function·getSockName():SocketAddress;-
function·getPeerName():SocketAddress;-
function·asStream():Stream;-

```

Eval

```

static·var·w_pipe_init:(Loop,·Bool)→Dynamic·=·neko.Lib.load("uv",·"w_pipe_ini
static·var·w_pipe_open:(Dynamic,·Int)→Void·=·neko.Lib.load("uv",·"w_pipe_open
static·var·w_pipe_accept:(Loop,·Dynamic)→Dynamic·=·neko.Lib.load("uv",·"w_pip
static·var·w_pipe_bind_ipc:(Dynamic,·neko.NativeString)→Void·=·neko.Lib.load(
static·var·w_pipe_connect_ipc:(Dynamic,·neko.NativeString,·Dynamic→Void)→Voi
static·var·w_pipe_pending_count:(Dynamic)→Int·=·neko.Lib.load("uv",·"w_pipe_p
static·var·w_pipe_accept_pending:(Loop,·Dynamic)→Dynamic·=·neko.Lib.load("uv"
static·var·w_pipe_getsockname:(Dynamic)→neko.NativeString·=·neko.Lib.load("uv
static·var·w_pipe_getpeername:(Dynamic)→neko.NativeString·=·neko.Lib.load("uv
static·var·w_pipe_write_handle:(Dynamic,·neko.NativeString,·Stream,·Dynamic→V
static·var·w_pipe_stream:(Dynamic)→Stream·=·neko.Lib.load("uv",·"w_pipe_strea

```

Neko

```

@:hlNative("uv",·"w_pipe_init")·static·function·w_pipe_init(_:Loop,·_:Bool):Na
@:hlNative("uv",·"w_pipe_open")·static·function·w_pipe_open(_:Native,·_:Int):V
@:hlNative("uv",·"w_pipe_accept")·static·function·w_pipe_accept(_:Loop,·_:Nati
@:hlNative("uv",·"w_pipe_bind_ipc")·static·function·w_pipe_bind_ipc(_:Native,·
@:hlNative("uv",·"w_pipe_connect_ipc")·static·function·w_pipe_connect_ipc(_:Na
@:hlNative("uv",·"w_pipe_pending_count")·static·function·w_pipe_pending_count(
@:hlNative("uv",·"w_pipe_accept_pending")·static·function·w_pipe_accept_pendin
@:hlNative("uv",·"w_pipe_getsockname")·static·function·w_pipe_getsockname(_:Na
@:hlNative("uv",·"w_pipe_getpeername")·static·function·w_pipe_getpeername(_:Na
@:hlNative("uv",·"w_pipe_write_handle")·static·function·w_pipe_write_handle(_:
@:hlNative("uv",·"w_pipe_stream")·static·function·w_pipe_stream(_:Native):Stre

```

HashLink

Eval

```
function·new(ipc:Bool);-
function·open(fd:Int):Void;-
function·connectIpc(path:String,·cb:Callback<NoData>):Void;-
function·bindIpc(path:String):Void;-
function·accept():Pipe;-
function·writeHandle(data:Bytes,·handle:eval.uv.Stream,·cb:Callback<NoData>):V
function·pendingCount():Int;-
function·acceptPending():PipeAccept;-
function·getSockName():SocketAddress;-
function·getPeerName():SocketAddress;-
function·asStream():Stream;-
```

Neko

```
static·var·w_pipe_init:(Loop,·Bool)→Dynamic·=-·neko.Lib.load("uv",·"w_pipe_ini
static·var·w_pipe_open:(Dynamic,·Int)→Void·=-·neko.Lib.load("uv",·"w_pipe_open
static·var·w_pipe_accept:(Loop,·Dynamic)→Dynamic·=-·neko.Lib.load("uv",·"w_pip
static·var·w_pipe_bind_ipc:(Dynamic,·neko.NativeString)→Void·=-·neko.Lib.load(
static·var·w_pipe_connect_ipc:(Dynamic,·neko.NativeString,·Dynamic→Void)→Voi
static·var·w_pipe_pending_count:(Dynamic)→Int·=-·neko.Lib.load("uv",·"w_pipe_p
static·var·w_pipe_accept_pending:(Loop,·Dynamic)→Dynamic·=-·neko.Lib.load("uv"
static·var·w_pipe_getsockname:(Dynamic)→neko.NativeString·=-·neko.Lib.load("uv
static·var·w_pipe_getpeername:(Dynamic)→neko.NativeString·=-·neko.Lib.load("uv
static·var·w_pipe_write_handle:(Dynamic,·neko.NativeString,·Stream,·Dynamic→V
static·var·w_pipe_stream:(Dynamic)→Stream·=-·neko.Lib.load("uv",·"w_pipe_strea
```

HashLink

```
@:hlNative("uv",·"w_pipe_init")·static·function·w_pipe_init(_:Loop,·_:Bool):Na
@:hlNative("uv",·"w_pipe_open")·static·function·w_pipe_open(_:Native,·_:Int):V
@:hlNative("uv",·"w_pipe_accept")·static·function·w_pipe_accept(_:Loop,·_:Nati
@:hlNative("uv",·"w_pipe_bind_ipc")·static·function·w_pipe_bind_ipc(_:Native,·
@:hlNative("uv",·"w_pipe_connect_ipc")·static·function·w_pipe_connect_ipc(_:Na
@:hlNative("uv",·"w_pipe_pending_count")·static·function·w_pipe_pending_count(
@:hlNative("uv",·"w_pipe_accept_pending")·static·function·w_pipe_accept_pendin
@:hlNative("uv",·"w_pipe_getsockname")·static·function·w_pipe_getsockname(_:Na
@:hlNative("uv",·"w_pipe_getpeername")·static·function·w_pipe_getpeername(_:Na
@:hlNative("uv",·"w_pipe_write_handle")·static·function·w_pipe_write_handle(_:
@:hlNative("uv",·"w_pipe_stream")·static·function·w_pipe_stream(_:Native):Stre
```

# ammer

```
class Foobar extends ammer.Library<"foobar"> {  
  public static function repeat(word:String, count:Int):String;  
}
```

```
class Main {  
  public static function main():Void {  
    trace(Foobar.repeat("hello", 3));  
  }  
}
```

```
--library ammer  
-D ammer.lib.foobar.include=include  
-D ammer.lib.foobar.library=lib  
--main Main  
--hl out.hl
```

# ammer

Under the hood:

- Types mapped to target-native analogues

- Functions accessed via automatically generated FFI

- A Makefile is generated that handles native compilation (hdll, ndll)

# ammer

API features:

- Functions

- Header defines

- Opaque types (abstract pointers)

More planned, of course, e.g.:

- Callbacks

- Structs



# ammer

Targets supported:

- Eval \*

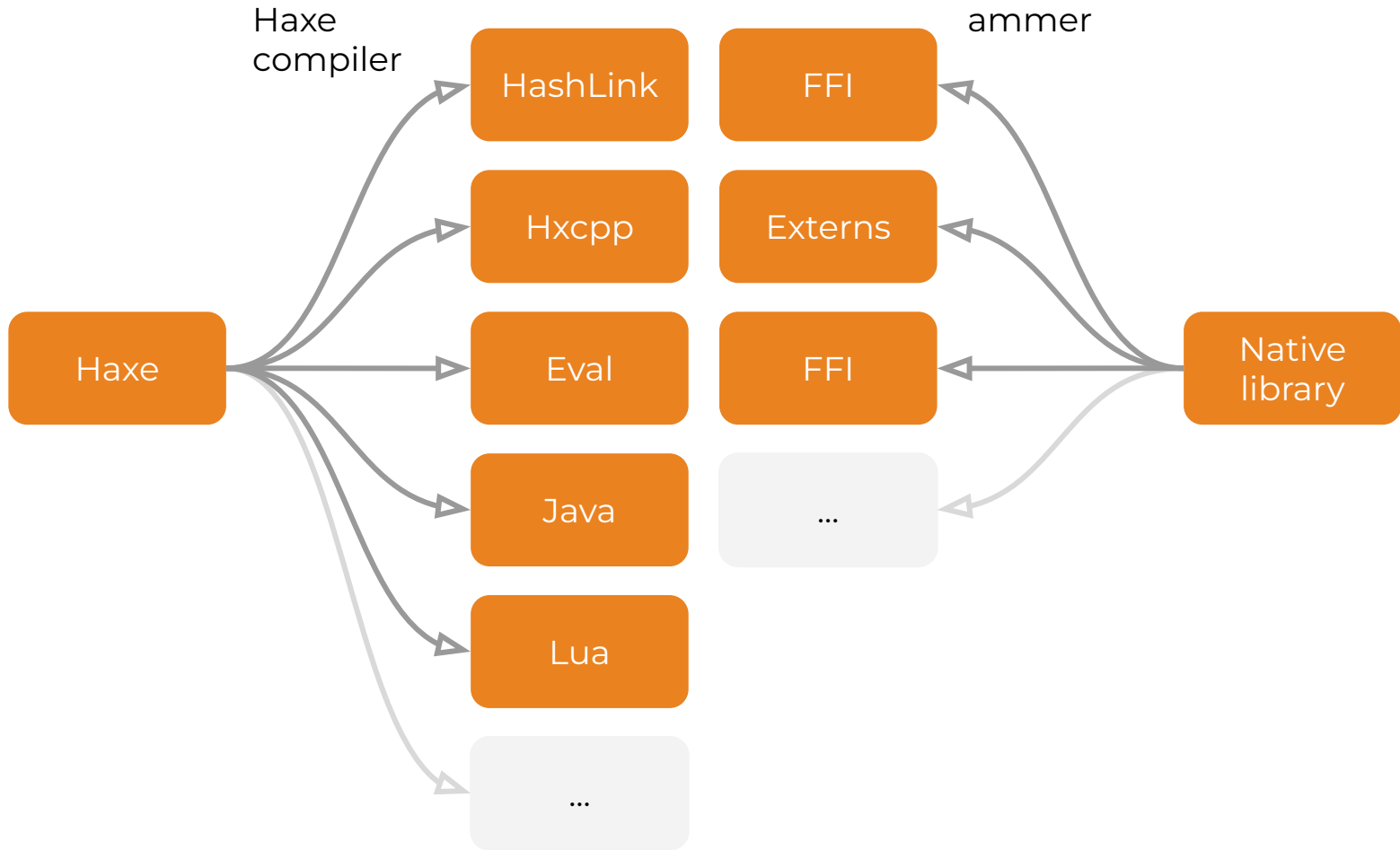
- HashLink

- Hxcpp

Targets planned:

- All sys targets!

[github.com/Aurel300/ammer](https://github.com/Aurel300/ammer)



# demo

ammer in action!

Lua!

SDL?

Libuv?

OpenAL?

Dear-imgui?

# Future work

HashLink GC

Thank you.