Native Deps a pain? WebAssembly can help!

Gavin Hayes
Senior Software Engineer at Dylibso

Introduction

dylibso

observe

EXTISM

xtp

Reasons to extend Perl

- The functionality isn’t implemented in Perl
- Access to APIs not available in Perl
- Speed
- Code reuse

Common ways of extending Perl

- XS - eXtendable Subroutine
- FFI (Foreign Function Interface) at runtime

Extending Perl with XS

"XS is less of an API and more of the guts of perl splayed out to do whatever you want. That may at times be very powerful, but it can also be a frustrating exercise in hair pulling." - `perldoc FFI::Platypus`

FFI - Foreign Function Interface

```
use FFI::Platypus 2.00;

# for all new code you should use api => 2
my $ffi = FFI::Platypus->new(
  api => 2,
  lib => undef, # search libc
);

# call dynamically
$ffi->function( puts => ['string'] => 'int' )->call("hello world");

# attach as a xsub and call (much faster)
$ffi->attach( puts => ['string'] => 'int' );
puts("hello world");
```

WebAssembly (Wasm)

```
(module
  (func $add (param $lhs i32) (param $rhs i32) (result i32)
    local.get $lhs
    local.get $rhs
    i32.add)
  (export "add" (func $add))
)
```

Extending Perl with Wasm

- The functionality isn’t implemented in Perl
- ~~Access to APIs not available in Perl~~
- Speed
- Code reuse

Building to WebAssembly

```
"$WASI_SDK_PATH/bin/clang" \
"--sysroot=$WASI_SDK_PATH/share/wasi-sysroot" \
"--target=wasm32-wasi" -o library.wasm library.c
```

```
cargo build --target wasm32-wasi --release
```

Building to WebAssembly

Building to WebAssembly

```
die('OS unsupported');
```

CPAN Testers Matrix

Result
FAIL
UNKNOWN

Using WebAssembly in Perl

```
use Wasm
  -api => 0,
  -wat => q{
    (module
      (func (export "add") (param i32 i32) (result i32)
       local.get 0
       local.get 1
       i32.add)
    )
  }
;
print add(1,2), "\n";  # 3
```

Example: string passing (Guest/Wasm side)

```
func count_vowels(input string) string {
    var count uint = 0
    for i := 0; i < len(input); i++ {
        count += is_vowel(input[i])
    }
    return fmt.Sprintf("%d", count)
}
```

Example: string passing (Guest/Wasm side)

```
//export count_vowels
func _count_vowels(ptr, size uint32) (ptrSize uint64) {
    name := ptrToString(ptr, size)
    g := count_vowels(name)
    ptr, size = stringToLeakedPtr(g)
    return (uint64(ptr) << uint64(32)) | uint64(size)
}
```

Example: string passing (Host/Perl side)

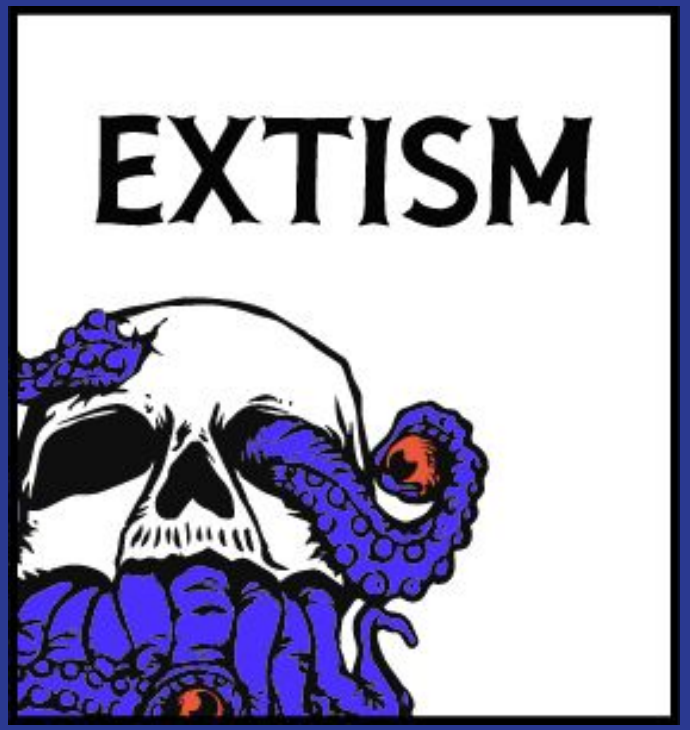
```
my $input = "hello";
say "input $input";
my $inptr = CountVowels::malloc(4);
store_string($CountVowels::memory->address + $inptr, $input);
my $outptrsize = CountVowels::count_vowels($inptr, length($input));
my $outsize = $outptrsize & 0xFFFFFFFF;
my $outptr = $outptrsize >> 32;
my $output = load_string($CountVowels::memory->address + $outptr, $outsize);
say "output $output";
```

Example: string passing (Host/Perl side)

- Failure to check the return value of malloc
- Out of bounds writing
- Leaking memory due to failure to call free

```
my $input = "hello";
say "input $input";
my $inptr = CountVowels::malloc(4);
store_string($CountVowels::memory->address + $inptr, $input);
my $outptrsize = CountVowels::count_vowels($inptr, length($input));
my $outsize = $outptrsize & 0xFFFFFFFF;
my $outptr = $outptrsize >> 32;
my $output = load_string($CountVowels::memory->address + $outptr, $outsize);
say "output $output";
```

Extism



Example: string passing (Guest/Wasm side w/ Extism)

```
//export count_vowels
func _count_vowels() int32 {
    input := pdk.Input()
    inputString := string(input)
    outputString := count_vowels(inputString)
    output := []byte(outputString)
    pdk.Output(output)
    return 0
}
```

Example: string passing (Host/Perl side w/ Extism)

```
my $wasm = do { local(@ARGV, $/) = 'count_vowels_extism.wasm'; <> };
my $plugin = Extism::Plugin->new($wasm, {wasi => 1});
my $input = "hello";
say "input $input";
my $out = $plugin->call('count_vowels', $input);
say "extism output $out";
```

What is Extism?

- Cross language framework for building with WebAssembly
  - Supports running WebAssembly in 15+ languages
  - Supports 8+ writing plugins in 8+ languages
- Easy ABI, each Extism functions always take a buffer of input data and return a buffer of output data, enabling easy string passing and avoiding manual memory management.
- Provides configuration management, resource limiting, and persistent variables, more

Extism Perl SDK

- Alien::libextism
- Extism

Extism Perl SDK

- Alien::libextism
- Extism
  - Extism::XS
  - Extism

What else is WebAssembly useful for?

Plugin System!

Example: Pluggable Audio Decoder

Interface:

- Open audio files
- Retrieve audio metadata
- Decode audio data

Plugin Interface:

- Open audio data
- Retrieve audio metadata
- Decode audio data

Example: Pluggable Audio Decoder (Host/Perl side)

```
sub new {
    my ($name, $filename, $plugins) = @_;
    my $audiodata = do { local(@ARGV, $/) = $filename; <> };
    foreach my $plugin (@$plugins) {
        say "trying plugin $plugin";
        my $wasm = do { local(@ARGV, $/) = $plugin; <> };
        my ($extism, $errmsg) = Extism::Plugin->new($wasm, {wasi => 1});
        $extism or croak $errmsg;
        my $decoder = $extism->call('open_memory', $audiodata) or next;
        return bless {'plugin' => $extism, 'decoder' => $decoder}, $name;
    }
    undef;
}
```

Example: Pluggable Audio Decoder (Host/Perl side)

```perl
sub get_metadata {
    my ($self) = @_;
    my $metajson = $self->{plugin}->call('get_metadata', $self->{decoder});
    $metajson or die "Failed to retrieve metadata";
    decode_json($metajson);
}

sub decode {
    my ($self) = @_;
    $self->{plugin}->call('decode', $self->{decoder});
}
```

Example: Pluggable Audio Decoder (Flac decoder)

```c
int32_t EXTISM_EXPORTED_FUNCTION(open_memory) {
  size_t input_size;
  void *input = extism_load_input_dup(&input_size);
  if (!input) {
    return 1;
  }
  drflac *decoder = drflac_open_memory(input, input_size, NULL);
  if (!decoder) {
    return 2;
  }
  extism_output_buf(&decoder, sizeof(decoder));
  // input is intentionally not free'd as it must stay alive as long
  // as the decoder
  return 0;
}
```

Example: Pluggable Audio Decoder (Flac decoder)

```c
int32_t EXTISM_EXPORTED_FUNCTION(get_metadata) {
  drflac *decoder;
  if (extism_input_length() != sizeof(decoder)) {
    return 1;
  }
  extism_load_input(0, &decoder, sizeof(decoder));
  char buf[128];
  snprintf(buf, sizeof(buf),
           "{\"sample_rate\": %u, \"bit_depth\": %u, \"channels\": %u}",
           decoder->sampleRate, decoder->bitsPerSample, decoder->channels);
  extism_output_buf_from_sz(buf);
  return 0;
}
```

Example: Pluggable Audio Decoder (Flac decoder)

```
int32_t EXTISM_EXPORTED_FUNCTION(decode) {
  drflac *decoder;
  if (extism_input_length() != sizeof(decoder)) {
    return 1;
  }
  extism_load_input(0, &decoder, sizeof(decoder));
  void *buf = malloc(decoder->totalPCMFrameCount * decoder->channels * 2);
  if (buf == NULL) {
    return 2;
  }
  uint64_t framesRead =
      drflac_read_pcm_frames_s16(decoder, decoder->totalPCMFrameCount, buf);
  extism_output_buf(buf, framesRead * decoder->channels * 2);
  free(buf);
  return 0;
}
```

Example: Pluggable Audio Decoder (Wav decoder)

```c
int32_t EXTISM_EXPORTED_FUNCTION(open_memory) {
  size_t input_size;
  void *input = extism_load_input_dup(&input_size);
  if (!input) {
    return 1;
  }
  drwav *decoder = malloc(sizeof(*decoder));
  if (!decoder) {
    return 2;
  }
  if (!drwav_init_memory(decoder, input, input_size, NULL)) {
    return 3;
  }
  extism_output_buf(&decoder, sizeof(decoder));
  // input is intentionally not freed so it stays alive with the decoder
  return 0;
}
```

Example: using Pluggable Audio Decoder

```
my $input_audio = $ARGV[0] // die "no input audio provided";
my $plugins = ['flac_decoder.wasm', 'wav_decoder.wasm'];
my $decoder = Extism::AudioDecoder->new($input_audio, $plugins);
$decoder or die "Failed to open decoder";
my $metadata = $decoder->get_metadata() or die "Failed to load metadata";
say "metadata:";
p $metadata;
my $pcm = $decoder->decode() // die "Failed to decode";
open(my $fh, '>', "$input_audio.pcm") or die "Failed to open out.pcm";
print $fh $pcm;
```

Questions?

https://github.com/extism/perl-sdk/