Dynamic Loading of Modules in WASM/Emscripten

Philipp Gloor

Table of Contents

Emscripten: Main Module, Side Modules I

In order to be able to dynamically load a WASM file they need to be compiled with specific options.

Following options can control how Emscripten creates the WebAssembly module:

- `-s MAIN_MODULE`
- `-s SIDE_MODULE`

The differences between the main module and the side modules:

Emscripten: Main Module, Side Modules II

- Main modules have the system libraries linked in
- Side modules are pure wasm files that contain LLVM bitcode that have no system libraries
- Compiling a main module adds a JavaScript file which sets up the Emscripten environment

Dynamic Loading I

Three different methods to dynamically link functions:

- dlopen/dlsym: Works the same as with native C code. Requires the side module to be available in the file system. Relies on C-Linkage or using mangled C++ names. Obtain dll handle with `dlopen` and link the functions with `dlsym`
- On startup: Define `Module.dynamicLibraries = [<list of wasm names>]`. Loads the wasm modules synchronously. Functions from modules need to be declared for compilation to work.
- On demand: Use `EM_ASM()` macro within C/C++.

Dynamic Loading II

Using `loadDynamicLibrary` from within C/C++ seems the best way

1. Loads WASM when really needed
2. Automatic linking (unlike dlopen)
3. Asynchronous

Passing values: Reference or Value