Shared Libraries in D

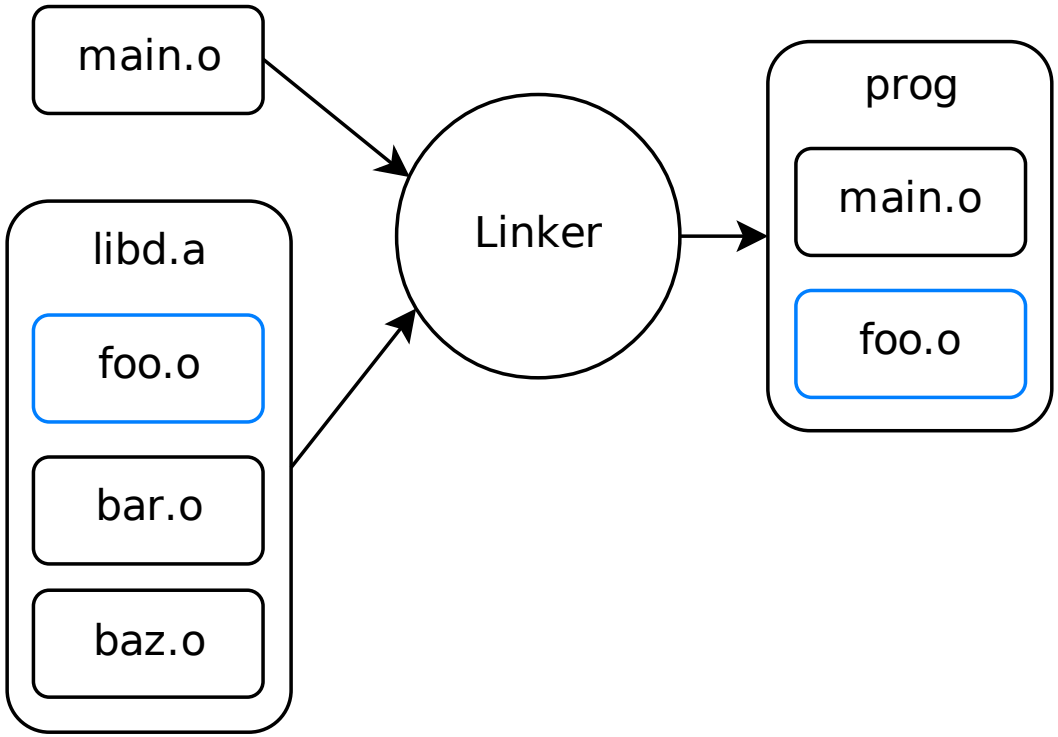
Martin Nowak

Agenda

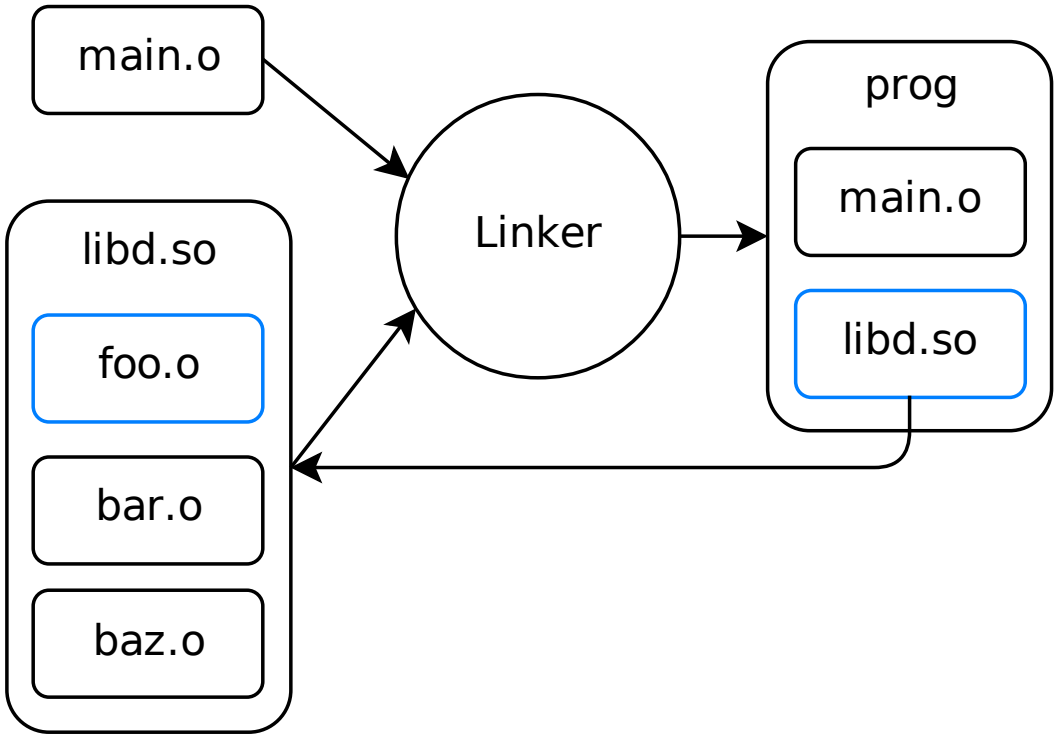
- Motivation
- Library Support
- Usage
- Details

Motivation

Linking against a static library



Linking against a shared library



Dynamic Linking

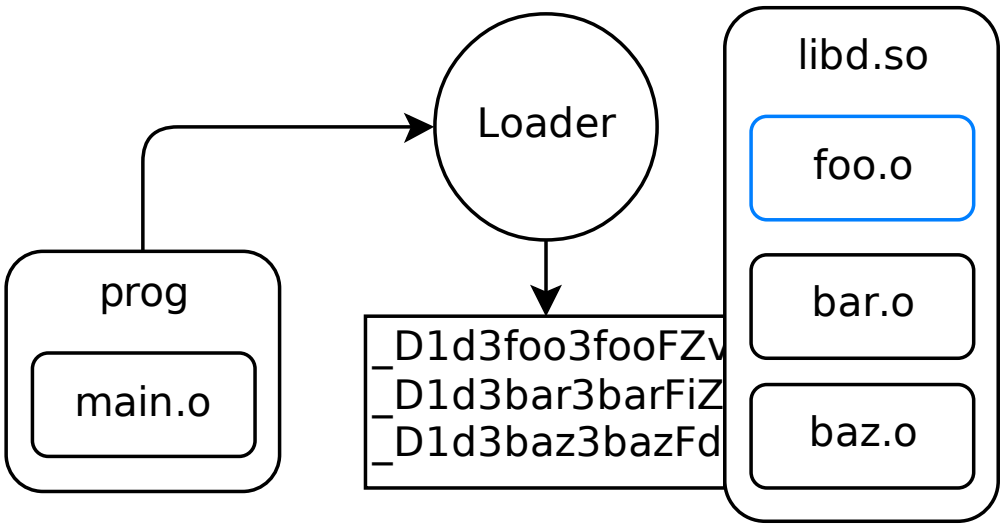
\+

- shared disc space
- shared memory
- bugfix updates
- faster linking

\-

- runtime overhead
- dependent executables

Runtime loading



Runtime loading

+

- load additional code at runtime
- optional dependencies

-

- runtime overhead

The Basics

Library support

Library support

currently

```
module core.runtime;

void* Runtime.loadLibrary(string path);
bool Runtime.unloadLibrary(void* lib);
```

Library support

currently

```
auto h = cast(HMODULE)enforce(Runtime.loadLibrary("foo.dll"));
scope (exit) Runtime.unloadLibrary(h);

auto bar = cast(void function())
     enforce(GetProcAddress(h, "D3foo3barFZv"));
bar();

auto baz = cast(string function(string))
     enforce(GetProcAddress(h, "D3foo3bazFAyaZAya"));
writeln(baz("hello"));
```

Library support

currently

```
auto h = enforce(Runtime.loadLibrary("libfoo.so"));
scope (exit) Runtime.unloadLibrary(h);

auto bar = cast(void function())
     enforce(dlsym(h, "_D3foo3barFZv"));
bar();

auto baz = cast(string function(string))
     enforce(dlsym(h, "_D3foo3bazFAyaZAya"));
writeln(baz("hello"));
```

Library support

Subtyping the platform handle

```
struct Library {
    void* _handle;
    alias _handle this;
}

Library Runtime.loadLibrary(string path) {
    return Library(dlopen(toStringz(path), RTLD_LAZY));
}

void Runtime.unloadLibrary(ref Library lib) {
    dlclose(lib._handle);
    lib._handle = null;
}
```

Library support

Adding methods

```
T loadFunc(T:FT*, FT)(string fqn) if (is(FT == function))
{
    immutable m = mangle!FT(fqn);
    return cast(T)dlsym(_handle, toStringz(m));
}


T loadFunc(T:FT*, string fqn, FT)() if (is(FT == function))
{
    static immutable m = mangle!FT(fqn);
    return cast(T)dlsym(_handle, m.ptr);
}
```

Library support

Adding methods

```
T* loadSym(T)(string fqn)
{
    immutable m = mangle!T(fqn);
    return cast(T*)dlsym(_handle, toStringz(m));
}


T* loadSym(T, string fqn)()
{
    static immutable m = mangle!T(fqn);
    return cast(T*)dlsym(_handle, m.ptr);
}
```

Library support

Loading a function address

```
auto lib = enforce(loadLibrary("libfoo.so"));
scope (exit) unloadLibrary(lib);

auto bar = enforce(lib.loadFunc!(void function())("foo.bar"));
bar();

auto baz = enforce(lib.loadFunc!(string function(string), "foo.baz")());
baz("hello");
```

- no mangled names
- hides platform differences
- kind of typesafe

Library support

Avoiding redundancy

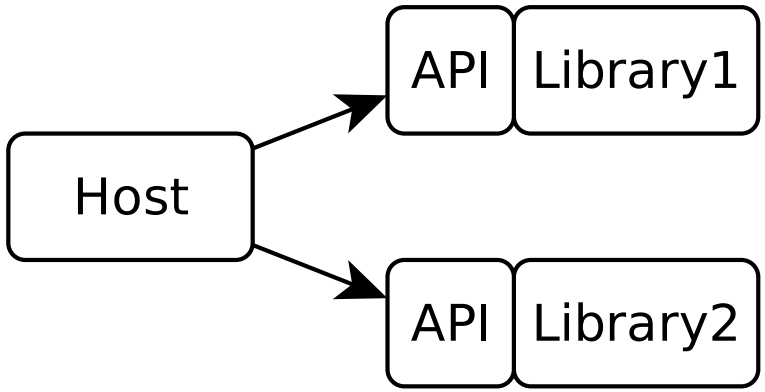
```
module foo; // foo.di
export void bar();
export string baz(string);
```

```
auto lib = enforce(loadLibrary("libfoo.so"));
scope (exit) unloadLibrary(lib);

import foo;
auto bar = enforce(lib.loadFunc!(foo.bar)());
bar();

auto baz = enforce(lib.loadFunc!(foo.baz)());
writeln(baz("hello"));
```

Usage

Usage

Loading a library at runtime



Usage

Loading a library at runtime

Library declares and defines interface

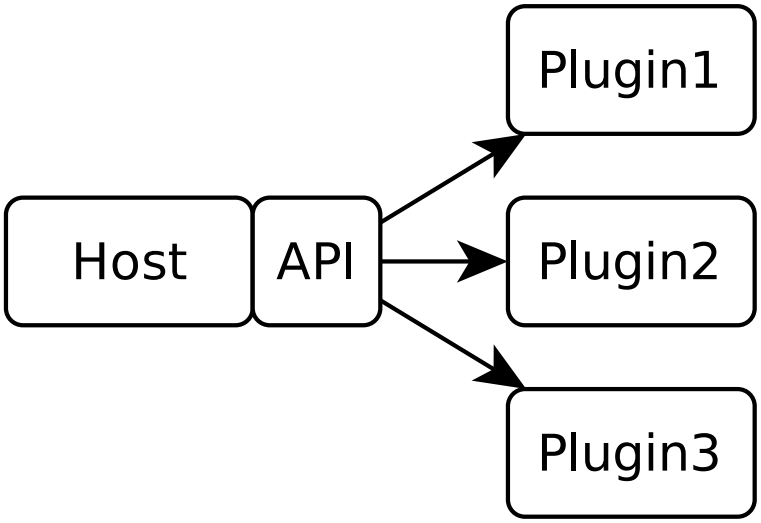
```
module http;
export string get(string url) { return shell("curl "~url); }
```

Host uses interface

```
if (auto lib = loadLibrary("libhttp.so"))
{
    scope (exit) unloadLibrary(lib);
    import http;
    auto get = enforce(lib.loadFunc!(http.get)());
    return get("http://dconf.org");
}
```

Usage

Plugin



Usage

Plugin

Host declares interface

```
module plugin; // plugin.di
export string process(string);
```

Plugin defines interface

```
module plugin;
export string process(string val) { return val; }
```

Host uses plugin

```
auto lib = enforce(loadLibrary(path));
auto process = lib.loadFunc!process();
```

Usage

Introspection

```
foreach (m; lib.modules) {
    writeln(m.name);
    writeln(m.localClasses);
    writeln(m.xgetMembers);
}
```

The Details

The Details

- loadLibrary initializes only the calling thread

The Details

- unloadLibrary needs to deal with stale objects

The Details

to be done

- refactor runtime initialization
- dynamic registration
- fix export

Thank You