Node.js の中身を見てみよう

src/node.cc のざっとした歩き方

IIJ 大津繁樹
ohtsuあっとiij.ad.jp

2011年11月9日
第1回 Node塾

言い訳・お詫び

- 昨晩、急きょ繰り上がり参加が決まり、今日午後で急いで資料作りました。
- 自分の過去のメモを中心に資料を作りましたが、**間違い**があるかもしれません。

- もし間違えに気付いた方はご連絡をお願いします。(_o_)

Node.js アーキテクチャ概要（v0.6.0)

zlib

openssl

http-parser

Node
User Modules

Node
Core Modules

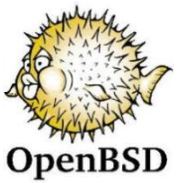
V8

libuv

c-ares

libev

libeio



Node.js ソースツリー構造概略

benchmark/ ベンチマークスクリプト
deps/ 外部ライブラリ群
    /http_parser
    /open_ssl
    /uv
    /v8
    /zlib
doc/ ドキュメント
lib/ ネイティブモジュール(*.js)
out/ ビルドディレクトリ
src/ 本体＋ビルトインモジュール（node_*.cc)
test/ テストスクリプト
tools/ ビルドツール

deps/zlib,openssl,
http-parser

zlib

openssl

http-parser

npmでインストール

Node
User Modules

（node_modules）

Node
Core Modules

src/
lib/

deps/v8

V8

deps/uv/src/unix

libuv

deps/uv

c-ares

libev

deps/uv/src/unix/ev

libeio

deps/uv/src/unix/eio

deps/uv/src/win

deps/uv/ares

opensolaris

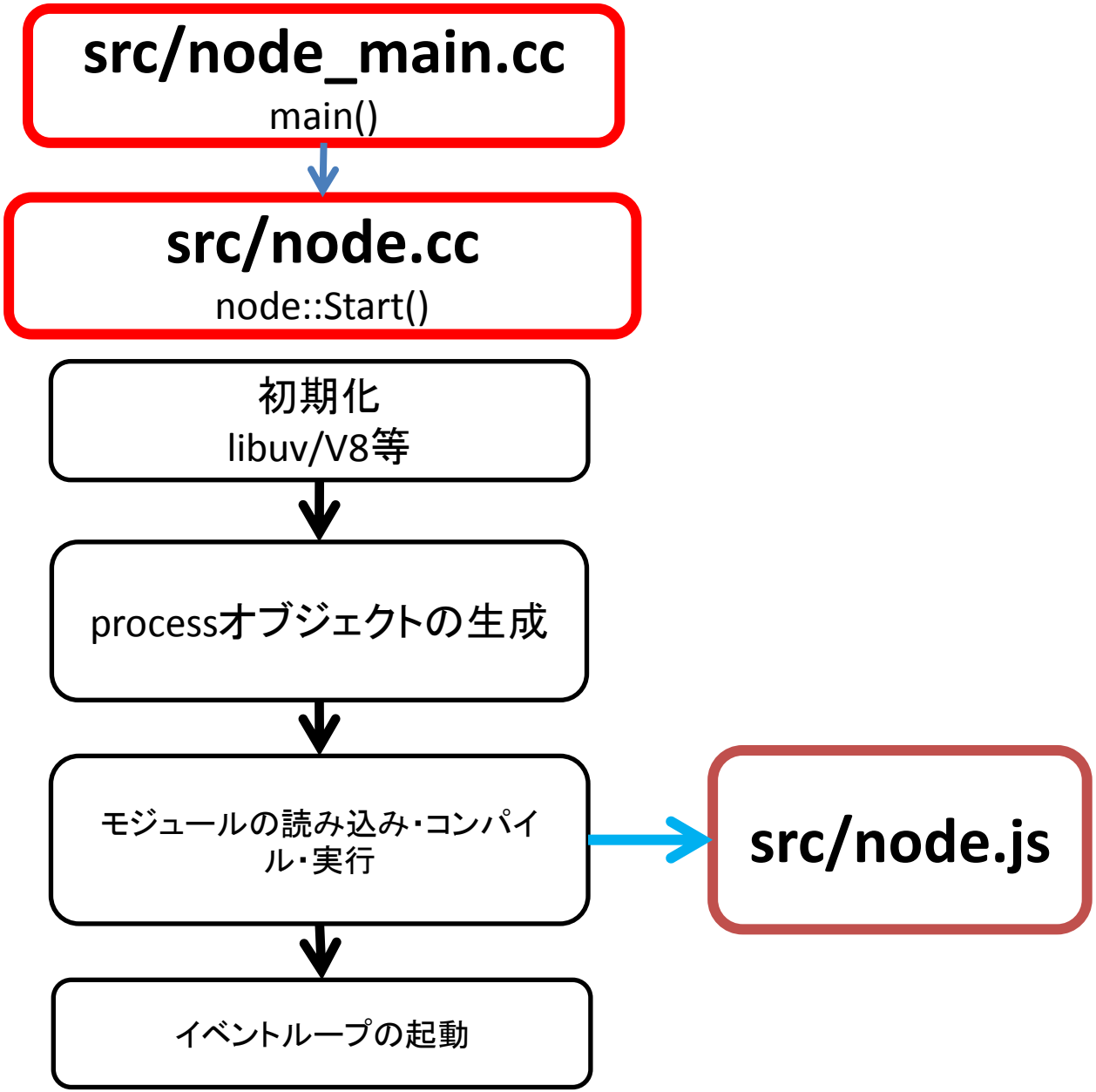
Mac

OpenBSD

FreeBSD

NetBSD

起動までのざぁ〜っとした流れ

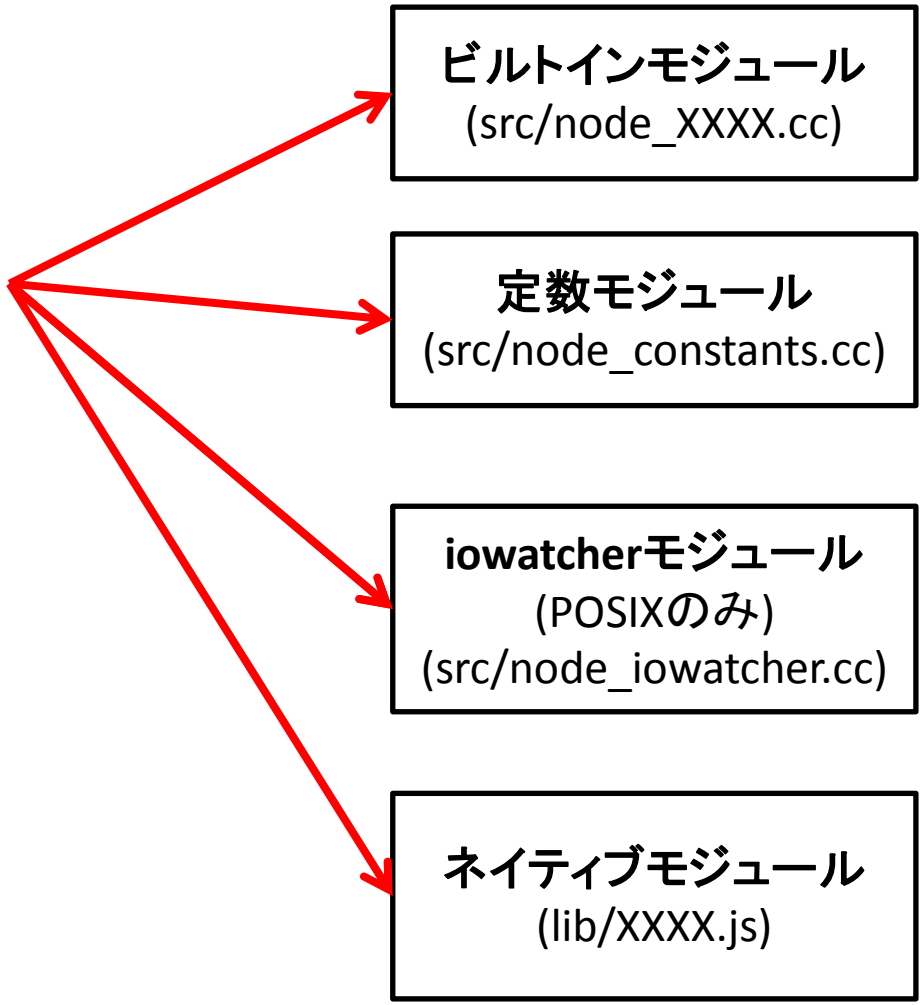


node::Startの処理 (src/node.cc:2455)

1. 主にlibuv系の初期化 (node.cc:2457)
   - argv = Init(argc, argv);
2. v8の初期化 (node.cc:2459)
   - v8::V8::Initialize();
3. V8コンテキストの生成 (node.cc:2463)
   - Persistent<v8::Context> context = v8::Context::New();
4. **processオブジェクトの生成 (node.cc:2466)**
   - **Handle<Object> process = SetupProcessObject(argc,argv);**
5. **オブジェクト、モジュールの読み込み、コンパイル、起動 (node.cc:2471)**
   - **Load(process);**
6. event loop の起動 () (node.cc:2478)
   - uv_run(); //イベント参照カウンターがなくなるまで無限ループを続ける
7. Exitイベントの発生 (node.cc:2480)
   - EmitExit(); //process.emit('exit') 発生

覚えておいた方がいいもの

process.binding()



C++ で書いたビルトインモジュールやJavaScript のネイティブモジュールとを橋渡しをする重要な関数

processオブジェクトの生成 (node.cc:2466)

特に process.binding の部分だけを説明

process.binding の処理 node.cc:1638

1. モジュール名の取得
   - 1641　Local<String> module = args[0]->ToString();
2. キャッシュオブジェクトの存在チェックと生成
   - 1645　if (binding_cache.IsEmpty()) {
   - 1646　　binding_cache = Persistent<Object>::New(Object::New());
   - 1647　}
3. モジュールがキャッシュされていたらそれを返して終了
   - 1651　if (binding_cache->Has(module)) {
   - 1652　　exports = binding_cache->Get(module)->ToObject();
   - 1653　　return scope.Close(exports);
   - 1654　}

processオブジェクトの生成 (node.cc:2466)

特に process.binding の部分だけを説明

4. 呼び出すモジュールが**ビルトインモジュール**の場合
   - 1662 if ((modp = get_builtin_module(*module_v)) != NULL) {
   - 1663 exports = Object::New();
   - 1664 modp->register_func(exports);
   - 1665 binding_cache->Set(module, exports);
   - 1666

- **ビルトインモジュール** : src/node_extentions.h にリストされているモジュールリスト
- src/node_foo.cc で作成されたモジュールは foo のV8オブジェクトハンドルを渡す。

(例）lib/fs.js:30 var binding = process.binding('fs');

src/node_fs.cc の node::InitFs(target) を実行し target(V8オブジェクトハンドル)を渡す。

processオブジェクトの生成 (node.cc:2466)

特に process.binding の部分だけを説明

5. 呼び出すモジュールが**コンスタント（定数）モジュール**の場合
   - 1667　} else if (!strcmp(*module_v, "constants")) {
   - 1668　　exports = Object::New();
   - 1669　　DefineConstants(exports);
   - 1670　　binding_cache->Set(module, exports);
   - 1671

- src/node_constants.cc の DefineConstants(target) を実行して target(V8オブジェクトハンドル)を返す。

（例）lib/fs.js:31　var constants = process.binding('constants');

lib/fs.js:167　　return constants.O_RDONLY;

processオブジェクトの生成 (node.cc:2466)

特に process.binding の部分だけを説明

6. 呼び出すモジュールが**io_watcherモジュール**の場合

- src/node_iowatcher.cc IOWatcher::Initialize(target) を実行し、target(V8オブジェクトハンドル)を返す。
  - 1672 #ifdef __POSIX__
  - 1673 } else if (!strcmp(*module_v, "io_watcher")) {
  - 1674 exports = Object::New();
  - 1675 IOWatcher::Initialize(exports);
  - 1676 binding_cache->Set(module, exports);
  - 1677 #endif

v.0.6.0 では利用例なし

processオブジェクトの生成 (node.cc:2466)

特に process.binding の部分だけを説明

7. 呼び出すモジュールが**ネイティブモジュール**の場合

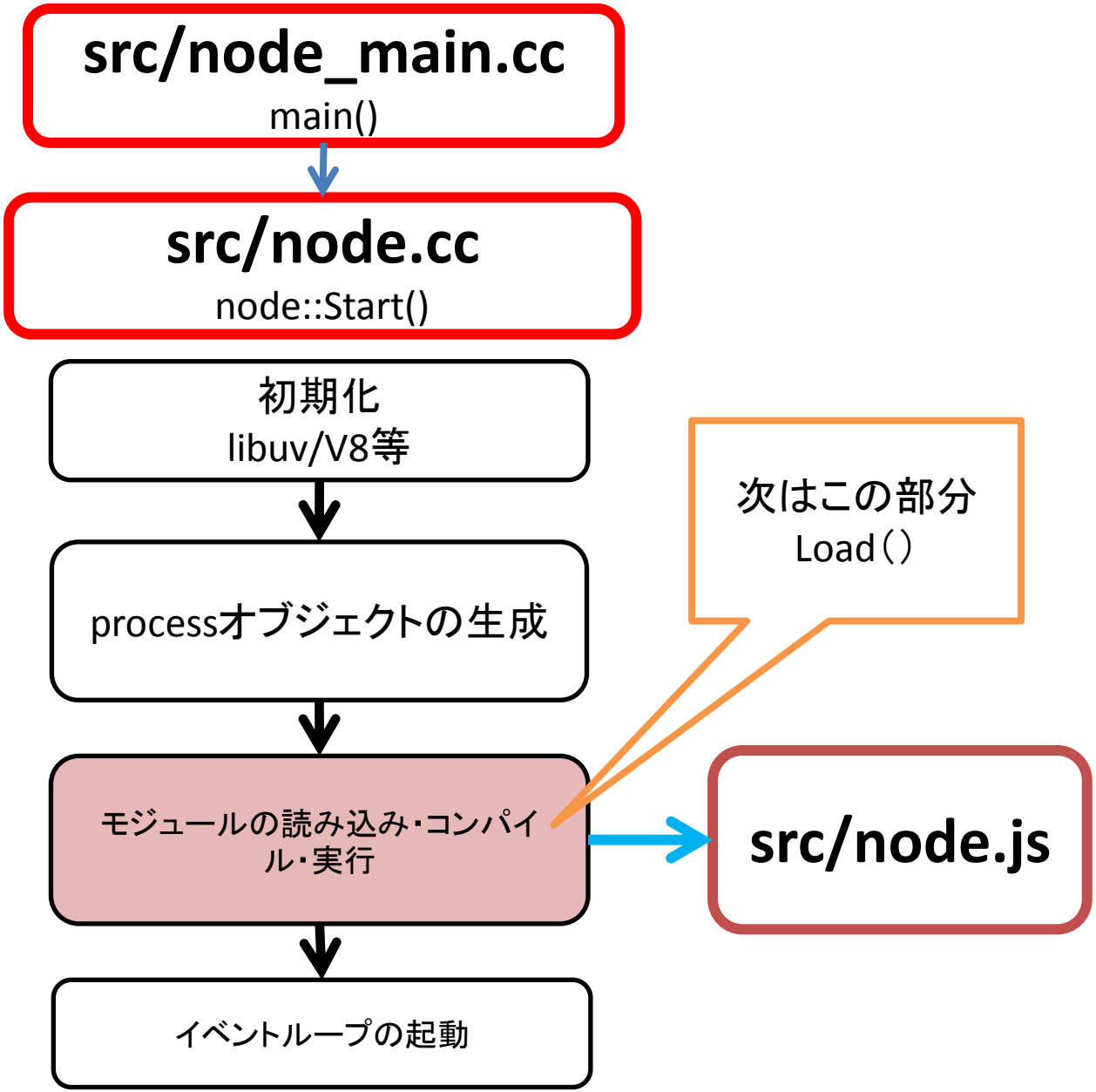
- src/node_javascript.cc の DefineJavaScript を使って **out/{Release,Debug}/src/node_natives.h** のデータ（元は lib/*.js)文字列を持つV8オブジェクトハンドルを返す。
  - 1679 } else if (!strcmp(*module_v, "natives")) {
  - 1680 exports = Object::New();
  - 1681 DefineJavaScript(exports);
  - 1682 binding_cache->Set(module, exports);

  (例) ビルトインと区別できるよう **NativeModule クラス**で扱っています。

  src/node.js: 475 NativeModule._source = process.binding('natives');

  src/node.js: 33 EventEmitter = NativeModule.require('events').EventEmitter;

8. 呼び出すモジュールオブジェクトを返す。→ 終了
   - return scope.Close(exports);


src/node_main.cc
main()
src/node.cc
node::Start()
初期化
libuv/V8等
processオブジェクトの生成
モジュールの読み込み・コンパイル・実行
イベントループの起動
次はこの部分
Load()
src/node.js

Load (src/node.jsのコンパイルと実行) (node.cc:1971)

1. **exit時の関数を登録**

   atexit(AtExit)

   AtExit → exit時には tty mode を reset する。

2. **src/node.js 関数をオブジェクト化**

```
Local<Value> f_value =
ExecuteString(MainSource(),IMMUTABLE_STRING("node.js"));
```

   - node_javascript.cc で MainSource が定義されている。

```
Handle<String> MainSource() {
  return BUILTIN_ASCII_ARRAY(node_native, sizeof(node_native)-1);
}
```

   - node_native は out/{Release,Debug}/src/node_natives.h にて定義

```
{ "node", node_native, sizeof(node_native)-1 },
```

   - node_natives.h はコンパイル時に node/wscript中の tools/js2c.py によって src/node.js lib/*.js をアスキーコード化(C-style Char Array)されている。

Load (src/node.jsのコンパイルと実行）(node.cc:1971)

3. **ExecuteString（src/node.jsオブジェクト取得）**
   - Local<v8::Script> script = v8::Script::Compile(source, filename); //src/node.jsのコンパイルしてスクリプト化
   - Local<Value> result = script->Run(); //src/node.jsのスクリプトを実行してオブジェクトを取得
   - return scope.Close(result); //src/node.jsのオブジェクトを返す。
4. **src/node.jsオブジェクトを関数へキャストする。**
   - Local<Function> f = Local<Function>::Cast(f_value);
5. **global と process オブジェクトの生成**
   - Local<Object> global = v8::Context::GetCurrent()->Global();
   - Local<Value> args[1] = { Local<Value>::New(process) }
6. **globalオブジェクト下で process オブジェクトを引数として src/node.js 関数を実行する。**
   - f->Call(global, 1, args);

7. そして src/node.js に続く・・・

```
(function(process) {
function startup() {
  EventEmitter = NativeModule.require('events').EventEmitter;
  ...
}
  NativeModule._source = process.binding('natives');
  ...
 startup();
});
```

おまけ

```
> node hoge.js


node.js:202
        throw e; // process.nextTick error, or 'error' event on first tick
                 ^
Error: Cannot find module '/home/ohtsu/node-v0.6.0/hoge.js'
    at Function._resolveFilename (module.js:334:11)
    at Function._load (module.js:279:25)
    at Array.<anonymous> (module.js:470:10)
    at EventEmitter._tickCallback (node.js:194:26)
```