emscripten/wasm ハンズオン

2021/5/25

村木太一

もくじ

- ハンズオン1[chrome dev toolsの基本]
- ハンズオン2[ブレークポイントの設定]
- ハンズオン3
  - emcmakeとemmakeの使い方
  - ファイルのパッケージ方法とembed-file
- ハンズオン4[embind]

参考サイト

- Chrome dev tools でのデバッグ方法
  - https://developer.chrome.com/blog/wasm-debugging-2020/
- emscripten
  - インストール方法 https://emscripten.org/docs/getting_started/downloads.html
  - パッケージ方法 https://emscripten.org/docs/porting/files/packaging_files.html
  - embind https://emscripten.org/docs/porting/connecting_cpp_and_javascript/embind.html

事前準備

[emscripten/c++/wasmハンズオン事前資料]
https://gist.github.com/mur6/665f381cc862e9dc823f9fcdc36c7fea

こちらの手順にそって、

1. emscriptenのインストール
2. デバッグ用chromeのセットアップ

が完了していること。

ハンズオン1[chrome dev toolsの基本]

参考資料

1. https://developer.chrome.com/blog/wasm-debugging-2020/#past
   の"The road so far"まで
2. 20210519_wasm_c++_chromeデバッグ方法
   に説明をまとめました。

ソース

1. https://github.com/mur6/wasm-emscripten-hands-on
   - フォルダ handson_01

ハンズオン1[chrome dev toolsの基本]

1. 適当なディレクトリを作成して下さい。
2. temp.c というファイルを作成して下さい。
3. 下記コマンドにて、c++のコードをwasmにコンパイルして下さい。
    - emcc -g temp.c -o temp.html
4. 下記コマンドにて、httpサーバを立ち上げて下さい。
    - python3 -m http.server 8000
5. セットアップが済んだchromeブラウザで、http://localhost:8000/main.html にアクセス。
6. Chrome DevToolsを起動して下さい。

ハンズオン2[ブレークポイントの設定]

参考資料

1. https://developer.chrome.com/blog/wasm-debugging-2020/#rich-types
   の"Rich type support"まで

ソース

1. https://github.com/mur6/wasm-emscripten-hands-on
   - フォルダ handson_02

ハンズオン2[ブレークポイントの設定]

以下でコンパイルして下さい。
(人によってはSDLのダウンロードが始まって時間がかかるかも。)

```
emcc -g mandelbrot.cc -o mandelbrot.html \
     -s USE_SDL = 2 \
     -s ALLOW_MEMORY_GROWTH = 1
```

- ブレークポイントの設定
- コンソール評価

のあたりを説明します。

ハンズオン3[emcmakeとembed-file]

参考資料

1. https://emscripten.org/docs/compiling/Building-Projects.html
    - emmakeについて
2. https://emscripten.org/docs/porting/files/packaging_files.html
    - 外部ファイルのパッケージング、埋め込みについて

ソース

1. https://github.com/mur6/wasm-emscripten-hands-on
    - フォルダ handson_03

ハンズオン3[emcmakeとembed-file]

以下の手順でコンパイルして下さい。

```
cd handson_03
cd build
emcmake cmake ..
cp ../hello_world.txt .
emmake make
```

- cmakeの前に `emcmake` を
- makeの前に `emmake` を

付けているのが分かるかと思います。

ハンズオン3[emcmakeとembed-file]

http://localhost:8000/filetest.html を開いたときの、
ブラウザでの表示結果:

```
Content=[Hello World!
]
```

ハンズオン4[embind]

参考資料

1. https://emscripten.org/docs/porting/connecting_cpp_and_javascript/embind.html

ソース

1. https://github.com/mur6/wasm-emscripten-hands-on
    - フォルダ handson_04

ハンズオン4[embind]

以下の手順でコンパイルして下さい。

```
cd handson_04
cd build
emcmake cmake ..
cp ../index.html .
emmake make
```

ハンズオン4[embind]

http://localhost:8000/index.html を開くと、
ブラウザのコンソールログに

```
lerp result: 1.5
```

と表示されます。

おまけ[The WebAssembly Binary Toolkit]

WebAssembly/wabt: The WebAssembly Binary Toolkit
wasmのデコンパイル等のいろいろなツールがある

- wasm2c
- wasm2wat