CMake を使用した Springhead Library のビルド

第 3.2 版 – 2021/09/15

本ドキュメントでは、CMake を用いて Springhead Library のセットアップからビルドまでを実施する流れについて説明します。unix においては、この方法が標準となります。

Windows については、配布キットに含まれているソリューションファイルを用いたビルドを推奨しており、cmake を使用してソリューションファイルを生成・ビルドする方法はあくまでオプションです。

Springhead Library のダウンロードについては “インストールガイド (Windows 版)” をご覧ください (ダウンロード方法は各プラットフォーム共通です)。

改 訂 履 歴

第 3.2 版　ライブラリタイプ指定の追加 (2.3 ビルドパラメータを変更する、3.1 CMake)
第 3.1 版　4 章の追加 (EmbPython のビルド)
第 3.0 版　全面改訂 (CMake に特化した記述に改訂)
第 2.0 版　全面改訂 及び Gitbook への移行
第 1.0 版　初版

目次

1　セットアップ

Springhead Library は、unix および Windows の両プラットフォームに対応しており、ビルド環境は CMake を用いて生成することができます (unix では Makefile、Windows では Visual Studio 用の solution/project file)。
この章では、Springhead Library のセットアップ方法について、unix と Windows のそれぞれについて説明します。

動作を確認しているプラットフォームは、以下のとおりです。

unix:
　Ubuntu 19.04.4 LTS (x86_64)
　cmake version 3.18.0
Windows:
　Windows 10 Enterprise
　Microsoft Visual Studio Community 2019, Version 16.9.3
　Windows SDK version 10.0.18362.0
　cmake version 3.18.1

本書では、Springhead Library をインストールしたディレクトリを "`.../Springhead`" と表記します。実際にインストールしたディレクトリに合わせて適宜読み替えてください。

セットアップでは、

- ビルドに必要なツールの確認
- 拡張版 `swig` のビルド (unix のみ)
- "`CMakeLists.txt`" の準備

などを実施します。
セットアップで得られた情報はセットアップファイル
　"`.../Springhead/core/src/setup.conf`"
に記録され、CMake 及びビルドの実行時に参照されます。また、セットアップが済んでいるか否かはこのファイルが存在するか否かで判断します。
なお、セットアップファイルを削除することでセットアップ前の状態を復元できます。

Springhead Library のビルドで使用するツールは次のものです。

`python`	複数のプラットフォームに統一して対応するため。 ビルドで使用する python script は python version 3 ベースで記述されています。拡張モジュール等を使用することで python 2.7.17 での動作は確認していますが、今後の修正において対応しきれないことも考えられます。Python 3 以降のバージョンが使える環境をお使いください。
`cmake`	Solution file/Makefile の生成を自動化するため。
`swig`	Springhead 用に拡張されたもの。EmbPython ライブラリをビルドするのに必要。unix の場合にはセットアップ時に生成します。Windows の場合には配布セットに含まれたものを使用します。
`gcc`, `gmake`	unix 環境のビルドツール。
`devenv`, `nmake`	Windows 環境における Visual Studio のビルドツール。Visual Studio のインストールキットに含まれています
`nkf`	encoding 変換のため。なくても大丈夫ですが、一部のメッセージが文字化けする可能性があります。

次の点にご注意ください。

- unix 環境において、デフォルトで gmake が見つからない場合 (`which gmake` で確認できます) には、gmake という名前で make にリンクを張ってください。
  例えば `cd /usr/bin; sudo ln -s ‘which make‘ gmake`
  なお、`make --version` としたときに `GNU Make 4.1` などと GNU Make の表示がでないときは、gmake のインストールが必要となります。
- unix の場合、必要なツールはすべて `PATH` に登録しておいてください。
- Windows 環境で複数の Visual Studio がインストールされている場合には、使用する Visual Studio のバージョンを選択することができます (後述)。
- Windows の場合、`devenv` 以外のツールは `PATH` に登録しておいてください。

1.1 unix でのセットアップ

1.1.1 初めてのセットアップ

ディレクトリ ".../Springhead/core/src" に移動して、スクリプト setup.sh を実行します。

```
> chdir .../Springhead/core/src
> ./setup.sh
found python (Version 3.4.2)
setup file ("setup.conf") not found.

currently available binaries are ...
-- checking python ...  found (version:  3.4.2)
-- checking gcc ...  found (version:  7.5.0)
-- checking swig ...  found (version:  2.0.4)
-- checking cmake ...  found (version:  3.18.0)
-- checking gmake ...  found (version:  4.1)
-- checking nkf ...  found (version:  2.1.4 (2015-12-12))

check result is ...
-- setup is required (reason:  setup file "setup.conf" not found,
"CMakeLists.txt" not found).

continue?  [y/n]:
```

ここで y と答えればセットアップファイルが作成されます。セットアップ作業はこれで終わりです。

1.1.2 再セットアップ

初めてのセットアップと同様、次のようにしてスクリプトを実行してください。

```
> chdir .../Springhead/core/src
> ./setup.sh
```

必要な環境に変更がなければ

```
  :
check result is ...
-- no need to execute 'setup'.
done
```

となり、スクリプトは終了します。

何らかの変更がある場合、たとえば

- 使用するツールのバージョン/PATH を変更する場合
- 拡張版 swig の再ビルドが必要となったとき

などの場合には

```
  :
check result is ...
-- setup is required (reason:  "swig" need to be rebuilt).

continue?  [y/n]:
```

などとなりますので、y で答えてください。必要な処理が実行され、セットアップファイルが更新されます。

> 何らかの理由で上記の “continue? [y/n]:” が表示されないとき、または強制的にセットアップファイルを再作成したいときには、setup コマンドに ‘-f’ オプションを付けて実行してください。

1.2 Windows でのセットアップ

1.2.1 初めてのセットアップ

ディレクトリ ".../Springhead/core/src" に移動して、スクリプト setup.bat を実行します。

```
> chdir .../Springhead/core/src
> setup.bat
-- found python:  C:/Python/python.exe

setup file ("setup.conf") not exists.

currently available binaries are ...
-- checking python ...  found (version:  3.5.4)
-- checking devenv ...  selection_number:  None
found (version:  15.9.28307.222)
-- checking swig ...  found (version:  2.0.4)
-- checking cmake ...  found (version:  3.18.1)

check result is ...
-- setup is required (reason:  "CMakeLists.txt" does not exist,
setup file "setup.conf" not found).
  :
-- (re)generating setup file ...
  :
```

セットアップ作業は以上です。

(注) devenv が複数見つかった場合は次のようになりますので、適切な番号を選択してください。

```
-- checking devenv ...  selection_number:  None
found multiple "devenv"
Select number (or '0' to try another one)
  (1) Visual Studio Community 2017 (15.9.28307.222)
  (2) Visual Studio Community 2019 (16.9.31129.286)
  (0) try another one
enter number:
```

1.2.2 再セットアップ

初めてのセットアップと同様、次のようにしてスクリプトを実行してください。

```
> chdir C:/Springhead/core/src
> setup.bat
```

必要な環境に変更がなければ

```
check result is ...
-- no need to execute 'setup'.
done
```

となり、スクリプトは終了します。

何らかの変更がある場合、たとえば

- 使用する Visual Studio のバージョンを変更する場合 (新しい Visual Studio をインストールした場合など)
- 使用するツールのバージョン/PATH を変更する場合

などの場合には

```
check result is ...
-- setup is required (reason:  "devenv" path differs,    (次の行に続く)
           "nmake" path differs).

continue?  [y/n]:
```

などとなりますので、y で答えてください。必要な処理が実行され、セットアップファイルが更新されます。

何らかの理由で上記の “continue? [y/n]:” が表示されないとき、または強制的にセットアップファイルを再作成したいときには、setup コマンドに -f オプションを付けて実行してください。

2　オプションの変更

この章では、

- 外部パッケージを利用する
- ライブラリとヘッダファイルのインストール先を指定する
- ビルド条件を変更する

などに対する設定方法について説明します。
配布されたデフォルト状態のままで構わない場合には、次の章“3 ビルド”に進んでください。

配布キットの中には、"`.../Springhead/core/src`" に次のファイルが含まれています。

`CMakeConf.txt.dist`	外部パッケージ導入の設定を定義する。 ヘッダファイル/ライブラリファイルのインストール先の設定を定義する。
`CMakeSettings.txt.dist`	ビルドパラメータの設定を行なう。

これらのファイルには各設定の既定値が設定されており、これらの値は cmake 実行時に参照され使用されます。
これらの既定値を変更する場合には、上記の該当するファイルを次のような名前でコピーし、コピーしたファイルを編集します (もとのファイルは変更しないでください)。cmake は実行時にこれらのコピーされたファイルを優先して参照します。

配布名	→	変更する名前
`CMakeConf.txt.dist`	→	`CMakeConf.txt`
`CMakeSettings.txt.dist`	→	`CMakeSettings.txt`

[参考]

変数 *variable* に値 *value* を設定するには `set(`*variable* "*value*"`)` とします。複数の値を設定するときは、空白文字で区切って並べたものを *value* とします。途中に空白やセミコロンを含まない文字列ならば引用符は省略できます。また、`${`*variable*`}` とすると他の変数の値を、`$ENV{`*variable*`}` とすると環境変数の値を参照できます。文字 '`#`' 以降はコメントです。

2.1 外部パッケージを利用する

別途インストールしたパッケージを使用する場合は、ファイル "CMakeConf.txt" の内容を次のように編集します。

変数 CMALE_PREFIX_PATH にパッケージを探索するパスを設定する。

```
set(CMAKE_PREFIX_PATH ".../somewhre/appropreate")
    #           (use absolute path)
    #           (multiple paths must be separated by 'newline' or 'semicolon')
)
```

実際に探索するパスは、<prefix> | <prefix>/(cmake|CMake) | <prefix>/<name>* | <prefix>/<name>*/(cmake|CMake) などです。ここで <prefix> は変数に設定したパス、<name> はパッケージ名を表します。詳細は cmake のドキュメントを参照してください。

2.2 ライブラリとヘッダファイルのインストール先を指定する

ライブラリファイルとヘッダファイルのインストール先を設定する場合は、ファイル "CMakeConf.txt" の内容を次のように編集します。

変数の設定は GUI からでも行なえます。ただし GUI での設定より "CMakeConf.txt" の設定の方が優先されます。また、GUI で CMAKE_INSTALL_PREFIX を空に設定した場合、または "CMakeConf.txt" で DO_NOT_GENERATE_INSTALL_TARGET を指定した場合には、インストールを行なうためのターゲットルールは生成されません。なお、"CMakeConf.txt" で CMAKE_INSTALL_PREFIX に空文字列を設定した場合には、GUI の場合とは異なり、デフォルトのインストールパス (Windows では "C:/Program Files"、unix では "/usr/local") が使用されます。

変数 CMAKE_INSTALL_PREFIX にインストール先を絶対パスで設定します。この設定をすることにより、find_package(Springhaed, REQUIRED) として Springhead Library を簡単に導入することができるようになります。

```
set(CMAKE_INSTALL_PREFIX ".../where/to/install")
    #           (use absolute path)
```

実際にファイルがインストールされるディレクトリは

```
config files   →   ${CMAKE_INSTALL_PREFIX}/Springhead
header files   →   ${CMAKE_INSTALL_PREFIX}/Springhead/include
library file   →   ${CMAKE_INSTALL_PREFIX}/Springhead/lib
```

です。これらのうち、ヘッダファイルとライブラリファイルのインストール先は次の変数を設定することで変更が可能です。絶対パスまたは ${CMAKE_INSTALL_PREFIX} からの相対パスで指定してください。

```
set(SPR_HEADERS_INSTALL_DIR "header files のインストール先")
set(SPR_LIBRARY_INSTALL_DIR "library file のインストール先")
```

2.3 ビルドパラメータを変更する

ファイル "CMakeSettings.txt" に定義されている変数のうち、次のものは変更することができます。

OOS_BLD_DIR	CMake の生成物を格納する作業場所 (ディレクトリ) の名称。 デフォルト値は build。
LIBTYPE	生成するライブラリのタイプ。静的ライブラリの場合は STATIC、 共有ライブラリの場合は SHARED を指定します。 Windows では STATIC のみ有効。unix でのデフォルトは SHARED。
VS_VERSION	Visual Studio のバージョン。devenv が PATH に含まれているならば、セットアップ時に自動的に設定されます。そうでない場合には、適切な値を設定してください。
COMP_FLAGS_ADD	追加するコンパイルフラグ。 既定値 COMP_FLAGS は "CMakeOpts.dist" に定義されています。
LINK_FLAGS_ADD	追加するリンクフラグ。 既定値 COMP_FLAGS は "CMakeOpts.dist" に定義されています。

その他の変数は変更しないでください。

3 ビルド

Springhead Library をビルドするには

1. CMake を使用して Makefile/Solution file を生成する
2. 生成された Makefile/Solution file を用いてビルドする

の 2 段階が必要です。

以下では、CMake の生成物 (*ビルドの生成物ではありません*) を格納する作業場所 (ディレクトリ) を "*build* " として話を進めます (作業場所の名前は任意で構いません)。

3.1 CMake

CMake には Configure と Generate の 2 段階があります。

コマンドプロンプトの場合は、1 回のコマンドで両方を実行できます。

```
> chdir .../Springhead
> mkdir build
> cmake -B build [generator] [libtype]
```

generator の 詳細は、コマンドプロンプトで `cmake --help` とすると確認できます。
generator を省略した場合のデフォルトは、unix では Makefile が選択されます。また Windows の場合には、インストールされている Visual Studio の最新バージョンがが選択されるようです。ただし、マシンアーキテクチャは自動的には判定されません。64 ビットマシンの場合には `-A x64` を指定してください。これを忘れると Visual Studio のプラットフォームが `x64` となりません。

---

generator の例

Windows:	`-G "Visual Studio 16 2019" -A x64`
unix:	`-G "Unix Makefiles"`

---

libtype は生成するライブラリのタイプを指定するパラメータで、次の何れかを指定します。

Windows:	`-D STATIC` (`SHREAD` は指定できません)
unix:	`-D STATIC` または `-D SHARED`

通常ライブラリタイプはファイル "`CMakeSettings.txt`" で指定しますが、パラメータ *libtype* の指定はそれより優先します。

cmake–gui を利用する場合は、まず、次の画面で Configure ボタンを押します。

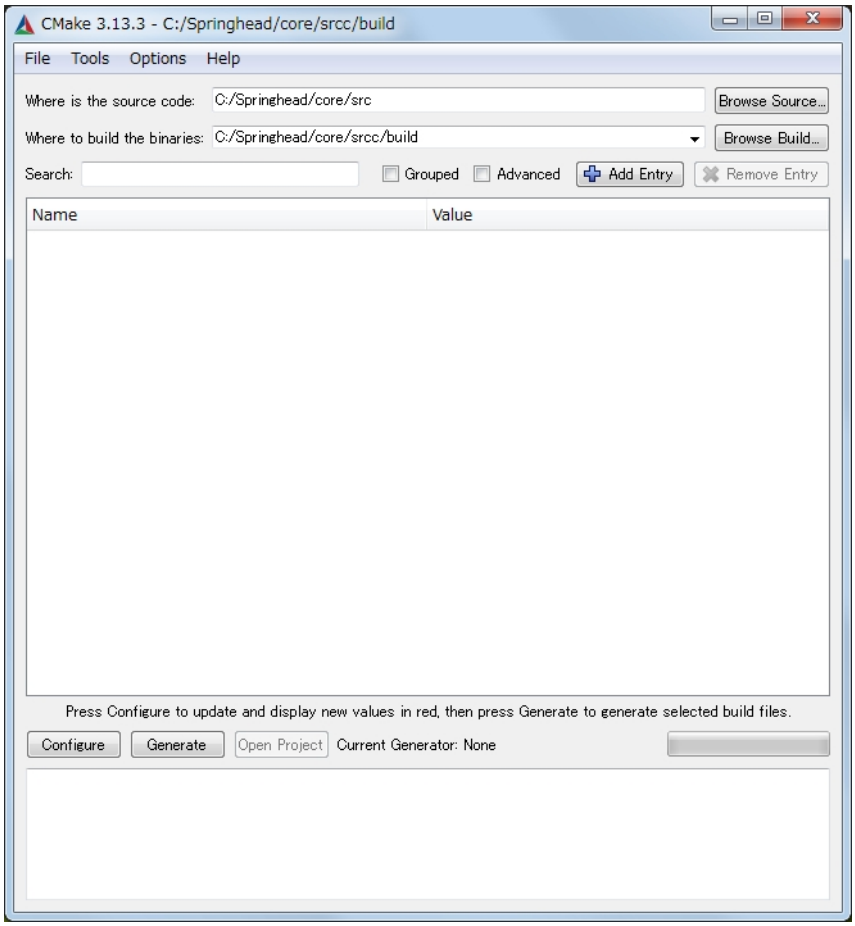

図 1　`cmake` configure

"*build*" ディレクトリがなければ作成するかどうかを尋ねられ、

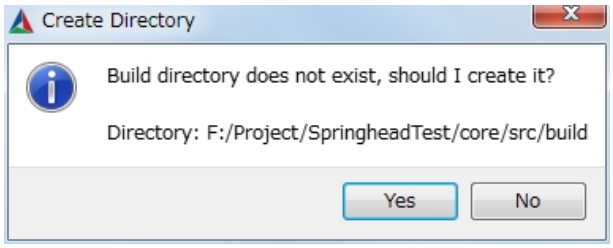

図 2　`cmake` configure

次に generator 指定画面となります。
最後に図 1 の Generate ボタンを押します。

以上で、"*build*" 以下に Makefile (unix の場合) または solution/project file (Windows の場合) が生成されたはずです。

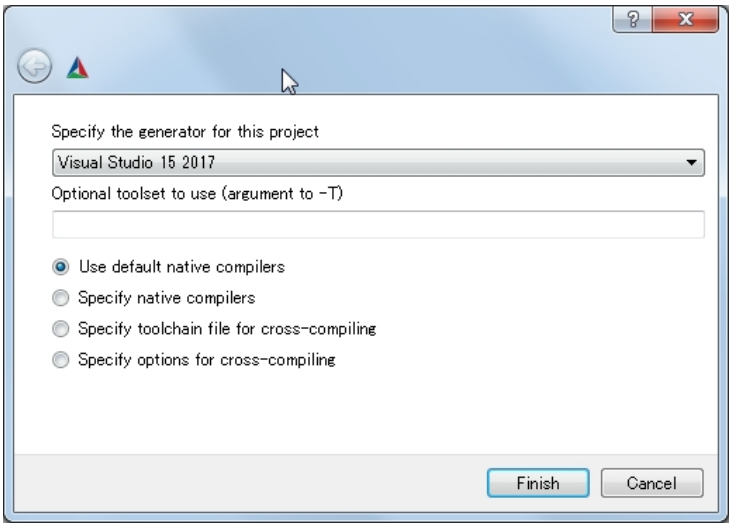

図 3　`cmake` configure

3.2　ビルド

ライブラリのビルドについては特に説明することはありません。

unix の場合

ディレクトリ *build* へ移動して make コマンドを実行してください。ライブラリファイルは "`.../Springhead/generated/lib`" に生成されます。
インストール先を指定した場合 (“2.2 インストールディレクトリの設定”参照) には、make コマンドの代わりに make install コマンドを実行してください。ライブラリファイル (及びヘッダファイル) は指定した場所にインストールされます。

Windows の場合

ディレクトリ *build* へ移動して "`Springhead.sln`" を Visual Studio で実行し、プロジェクト Springhead をビルドしてください。ライブラリファイルは
"`.../Springhead/generated/lib/<`*arch*`>`" に生成されます。<*arch*> はマシンのアーキテクチャに従い、 "`win64`" または "`win32`" のいずれかです。
インストール先を指定した場合 (“2.2 インストールディレクトリの設定”参照) には、プロジェクト Springhead のビルドに続けてプロジェクト INSTALL を (プロジェクトのみ) ビルドしてください。ライブラリファイル (及びヘッダファイル) は指定した場所にインストールされます。

Springhead Library を使用したプログラムの実行時に DLL が見つからないというエラーが発生した場合には、

32 ビット環境のときは —	"…/Springhead/dependency/bin/win32"
64 ビット環境のときは —	"…/Springhead/dependecny/bin/win64" と "…/Springhead/dependecny/bin/win32" の両方

にパスを通してください。Visual Studio から実行するときは、プログラムのプロパティを開き、[構成プロパティ]—[デバッグ]—[環境] に “path=上記のパス” とします。

4　EmbPython ライブラリのビルド

unix において EmbPython のライブラリをビルドする方法を説明します。Windows の場合については “インストールガイド 1.3 EmbPython のビルド” をご覧ください。

4.1　unix でのビルド

ディレクトリ "`.../Springhead/core/src/EmbPython`" に移動し、次のように cmake コマンド及び make コマンドを実行します。

```
> chdir .../Springhead/core/src/EmbPython
> cmake -B build -D STANDALONE=1 [-D LIBTYPE=STATIC]
> chdir build
> make
```

cmake の最後の引数 "`-D LIBTYPE=STATIC`" を指定したときは static library (`libEmbPython.a`) が、省略したときは shared library (`libEmbPython.so`) が、 "`.../Springhead/generated/lib`" に作成されます。

作成するライブラリファイルのインストール先を指定する場合は、配布ファイルセットの中にある "`.../Springhead/core/src/CMakeConf.txt.dist`" を "`CMakeConf.txt`" という名前でコピーしてインストール先を設定し、上記の make コマンドの代わりに make install コマンドを実行してください。
インストール先の設定方法は、“2.2 ライブラリとヘッダファイルのインストール先を指定する”をご覧ください。

```
> chdir .../Springhead/core/src/EmbPython
> cp ../CMakeConf.txt.dist CMakeConf.txt
> edit CMakeConf.txt
> cmake -B build -D STANDALONE=1 [-D LIBTYPE=STATIC]
> chdir build
> make install
```