

# シンプルで応用の効くmakefileとその解説


`makefile` は一度作るとそれ以降編集する機会が少なくなるので意外と真面目に考える人は少なく、ネット上でもまとまった情報は多くない。

Linux系OS上(正確に言うとGNU MakeとGCC)で複数の `C/C++` ソースファイルから1つの実行ファイルを作成(`make`)するための汎用的な `makefile` テンプレートを作った。名前はまだない。適宜ディレクトリ構成や設計などに従って `makefile` をカスタマイズする必要があると思うがそのベースにする。

## ■ このmakefileのいいところ

- コンパイル対象となるソースファイルをワイルドカードで指定できる。
- ヘッダファイル、ライブラリ、オブジェクトファイルなどコンパイル、リンクに関連するどのファイルが外部で変更されていてもきちんと 差分 コンパイルされる。
- `makefile`自体に説明書(この記事)がある。

## ■ `makefile`

[Gistはこちら](#) 。

```
COMPILER    = g++
CFLAGS      = -g -MMD -MP -Wall -Wextra -Winit-self -Wno-missing-
ifeq "$(shell getconf LONG_BIT)" "64"
    LDFLAGS =
else
    LDFLAGS =
endif
LIBS        =
INCLUDE     = -I./include
TARGET      = ./bin/$(shell basename `readlink -f .`)
SRCDIR      = ./source
ifeq "$(strip $(SRCDIR))" ""
    SRCDIR = .
endif
SOURCES     = $(wildcard $(SRCDIR)/*.cpp)
OBJDIR      = ./obj
ifeq "$(strip $(OBJDIR))" ""
    OBJDIR = .
endif
OBJECTS     = $(addprefix $(OBJDIR)/, $(notdir $(SOURCES:.cpp=.o))
DEPENDS     = $(OBJECTS:.o=.d)

$(TARGET): $(OBJECTS) $(LIBS)
    $(COMPILER) -o $@ $^ $(LDFLAGS)

$(OBJDIR)/%.o: $(SRCDIR)/%.cpp
    -mkdir -p $(OBJDIR)
    $(COMPILER) $(CFLAGS) $(INCLUDE) -o $@ -c $<

all: clean $(TARGET)

clean:
    -rm -f $(OBJECTS) $(DEPENDS) $(TARGET)

-include $(DEPENDS)
```

## ■ 使用方法

## ■ ■ make

基本的には `make` するだけで依存関係を考慮して差分コンパイルされる。ソースファイルはもちろん、ヘッダファイル、ライブラリなどが更新されている場合も自動的に検出して差分コンパイルされる。

## ■ ■ make all

強制的に全ソースをコンパイルしたい場合は `make all` する。このコマンドは全ての中間ファイル(オブジェクトファイル、依存関係ファイル)と実行ファイルを削除してから全ソースをコンパイルする。

## ■ ■ make clean

全ての中間ファイル(オブジェクトファイル、依存関係ファイル)と実行ファイルを削除する。

# ■ 初期設定状態のmakefile

下記のようなディレクトリツリーで初期設定の `makefile` を用いて `make` を実行した場合、以下の矢印(<)のようにファイルが生成される。同一のディレクトリに存在する全てのcppファイルがコンパイル(及びリンク)対象となる。

```
example
|-- makefile
|-- bin
|   |-- example    <- (TARGET) 実行ファイル
|-- include
|   |-- example.h
|-- obj            <- (OBJDIR) 中間ファイル生成先ディレクトリ
|   |-- example.d  <- (DEPENDS) 依存関係ファイル
|   |-- example.o  <- (OBJECTS) オブジェクトファイル
|-- source
|   |-- example.cpp
```

## ■ 解説

### ■ コンパイラの指定 ( COMPILER )

コンパイラは COMPILER の値を用いる。初期値は g++ 。C言語のみの場合は gcc に変更しても良いが基本的には g++ で問題ない。

### ■ コンパイルオプション ( CFLAGS )

コンパイルオプションとして CFLAGS の値を用いる。-D オプションによる #define の追加、最適化オプション、コードカバレッジ用の -coverage などを用いる場合はここに記述する。

以下は初期値の解説。

- -Wall -Wextra -Winit-self -Wno-missing-field-initializers
  - -Wall : コンパイルワーニングのレベルを最大にする。
  - -Wextra : 歴史的理由により -Wall を使用でも抑制される警告を抑制しない。つまり可能な限り全ての警告を出す。
  - -W\* (それ以外) : 詳しくは [Warning Options - Using the GNU Compiler Collection \(GCC\)](#) <sup>🔗</sup> を参照されたい。
- -g
  - デバッグオプション。
  - gdb でのデバッグを可能にする。
- -MMD -MP
  - ソースファイルの依存関係を中間ファイルに出力する。
  - 依存関係ファイルはソースファイル名の拡張子を .d に変更したものとなり、OBJDIR で指定したディレクトリに生成される。

- 。この依存関係ファイルは `makefile` 最後の `include` 文にてインクルードされることで依存しているヘッダファイル等が変更された場合に自動的に再コンパイルされるようになる。

## ■ リンクオプション ( `LDFLAGS` )

リンクオプションとして `LDFLAGS` の値を用いる。初期値は空。動的ライブラリをリンクする `-l` オプションを用いる場合はここに記述する。パスの通っていない動的ライブラリをリンクするならここにそのファイル名( `*.so` みたいな)を書いても良い。

## ■ ライブラリの指定 ( `LIBS` )

静的リンクするライブラリとして `LIBS` の値を用いる。初期値は空。静的ライブラリ( `*.a` )を用いる場合、空白区切りでファイル名を記述する。ここで指定したライブラリが更新された場合、`make` は再コンパイルが必要だと認識する。

## ■ インクルードパスの指定 ( `INCLUDE` )

インクルードパスとして `INCLUDE` の値を用いる。初期値は `-./include` 。ソースファイル中の `#include` ファイル検索パスに加えるパスを `-I` オプションにて指定する。`-I` オプションとディレクトリ名の間に空白を書くことはできない。複数ディレクトリを指定したい場合は `-I` オプションを空白区切りで複数指定する。

## ■ 実行ファイルの指定 ( `TARGET` )

実行ファイル名として `TARGET` の値を用いる。以下は初期値

`./bin/$(shell basename `readlink -f .`)` の解説。実行ファイルの生成先のディレクトリは `./bin`。生成される実行ファイル名は `$(shell basename `readlink -f .`)` である。これは `makefile` の存在するディレクトリの名前。

## ■ 中間ファイル生成先ディレクトリの指定 (`OBJDIR`)

中間ファイル生成先ディレクトリとして `OBJDIR` の値を用いる。初期値は `./obj`。このフォルダにオブジェクトファイル(`*.o`)や依存関係ファイル(`*.d`)が生成される。

## ■ ソースファイルの指定 (`SOURCES`)

コンパイル対象となるソースファイルとして `SOURCES` の値を用いる。初期値は `$(wildcard $(SRCDIR)/*.cpp)`。`SRCDIR` に存在する拡張子 `cpp` のファイル全てをコンパイル対象とすることを意味する。別の拡張子(`.c` など)に変更したい場合は、`makefile` 内の `cpp` を全て変更する。

## ■ オブジェクトファイルの指定 (`OBJECTS`)

オブジェクトファイルとして `OBJECTS` の値を用いる。以下は初期値

`$(addprefix $(OBJDIR)/, $(notdir $(SOURCES:.cpp=.o)))` の解説。オブジェクトファイルの生成先ディレクトリは `OBJDIR`。オブジェクトファイル名はソースファイル(`SOURCES`)の拡張子を `.o` に置換したもの。`OBJDIR` が空の場合は `makefile` と同一のディレクトリに生成される。

## ■ 依存関係ファイルの指定 (`DEPENDS`)

依存関係ファイルとして `DEPENDS` の値を用いる。初期値 `$(OBJECTS:.o=.d)` はオブジェクトファイルの拡張子を `.d` に置換したもの。

そんじゃーね。に花束を。

📅 2013-08-10    💬 コメント 2    🏷️ C++ (/tags/C++) 2    🏷️ makefile (/tags/makefile) 1

🏷️ gcc (/tags/gcc) 1    🏷️ Linux (/tags/Linux) 1

 Follow

いいね! 1 feed    45 ip    Bookmark 8 http    Tweet 2Fur G+1 1 io%2Frss.xml)

🔙 Jekyll 1.1系をWindowsに導入する

(/posts/2013/Setup-jekyll-1.1-to-windows)

要素の高さを揃える超軽量なjQueryプラグインjquery.tile.js 🔙

(/posts/2013/release-jquery-tile-js)

2 件のコメント    URIN HACK

1 ログイン ▼

♥️ オススメする 2    🔄 共有

評価順に並び替え ▼



コメントを投稿する...



青子守歌 • 2年前

こんにちは。こちらで作成されたMakefileを私の開発しているプロジェクト(<https://github.com/aokomoriuta...>)でも使用させていただきたいのですが、ライセンス等はどうなっていますでしょうか。ひとまずMakefile冒頭にthanks toを入れさせて頂いていますが、もし不十分でしたら、ご指摘いただけると助かります。よろしくお願いします。

^ | v • 返信 • 共有 >



URIN HACK 管理人 ➡ 青子守歌 • 2年前

ライセンスフリーですよー。

^ | v • 返信 • 共有 >

✉️ 更新を受取る    🌐 あなたのサイトにDisqusを追加    Disqusを追加    追加    🔒 非公開

## Recent Entries

2015-09-19

logrotate.bat - Windowsバッチで(指定サイズ指定を超えたら)ログローテート (/posts/2015/logrotate-batch-on-windows)

2015-04-05

C++で無名関数の関数ポインタを作る (/posts/2015/lambda-without-stdcpp)

2014-08-14

Jekyll 2.3.0をWindowsに導入 (/posts/2014/install-jekyll-2.3.0-to-windows)

2014-03-02

Jekyll 1.4.2をRuby 2.0.0+Windowsに導入 (/posts/2014/install-jekyll-1.4.2-to-windows)

2014-02-12

設置が超簡単でカスタマイズ性の高い吹き出しホバー用jQueryプラグインjquery.balloon.js (/posts/2014/jquery-balloon-js)

2013-08-11

要素の高さを揃える超軽量なjQueryプラグインjquery.tile.js (/posts/2013/release-jquery-tile-js)

2013-08-10

シンプルで応用の効くmakefileとその解説 (/posts/2013/simple-makefile-for-clang)

2013-08-04

Jekyll 1.1系をWindowsに導入する (/posts/2013/Setup-jekyll-1.1-to-windows)

2013-08-03

ブログをリニューアル (/posts/2013/Renewal)

## Tags

Linux (/tags/Linux)

**Ruby (/tags/Ruby)**

gcc (/tags/gcc)

JavaScript (/tags/JavaScript)

**Windows (/tags/Windows)**

Batch (/tags/Batch)

jQuery (/tags/jQuery)

GitHub

(/tags/GitHub)

C++ (/tags/C++)

**Jekyll (/tags/Jekyll)**

makefile (/tags/makefile)

Octopress

(/tags/Octopress)



## Repositories

**jquery.tile.js ([/jquery.tile.js](#))** Star 43

要素の高さを揃える超軽量なjQueryプラグイン

**jquery.balloon.js ([/jquery.balloon.js](#))** Star 28

設置もカスタマイズも超簡単な吹き出しホバー用jQueryプラグイン

**wers ([//github.com/urin/wers](#))** Star 2

Windows用Rubyバージョン管理ツール

(rbenvのWindows版)

**jquery.stretchable.js ([//github.com/urin/jquery.stretchable.js](#))**

要素を浮かせて伸ばすボタンを付けるjQueryプラグイン

**.vimrc ([//github.com/urin/.vimrc](#))**

Vim用設定ファイル



([//getbootstrap.com/](#))

Styled with Twitter Bootstrap 3.

([//jquery.com/](#))

Scripted with jQuery version 1.11.1.

([//jekyllrb.com/](#))

Powered by Jekyll version 2.3.0.

(//github.com/)

Hosted on GitHub.

(//git-scm.com/)

Managed with git.

---

© 2013 - 2016 うりん



(//github.com/urin)



(//twitter.com/urinhack)



(//www.facebook.com/URIN.HACK)