

minus9d's diary

2014-02-04

Makefileの書き方に関する備忘録 その2

linux

2

ブックマーク

この記事は続き記事です。目次→[Makefileの書き方に関する備忘録 - minus9dの日記](#)

変数

ユーザが編集したりコマンドラインでカスタマイズしたりしたくなるかもしれない変数は大文字、[makefile](#)内で閉じて使われる変数は小文字で書くのが慣例。

```
CC := gcc
sources = *.c
```

変数名の後の半角スペースに注意

以下のように、変数名の後にコメントを書くと、変数\$(LIBRARY) は "libio.a "となり、半角スペースを巻き込んでしまう。

```
LIBRARY = libio.a #comment
```

変数のタイプ

変数のタイプには2種類ある。

expanded variables

expanded variablesは、コロン付きのイコールで定義される変数。定義式を評価するときに、右辺の変数が即時展開される。

```
CC = g++
MAKE_DEPEND := $(CC) -M

all:
    @printf "$(MAKE_DEPEND)"

CC = gcc
```

例えば上の[makefile](#)を実行すると、2行目を読み込んだ時点で、変数\$(MAKE_DEPEND)は"g++ -M"に確定する。よってmake allとしたときの出力は

```
g++ -M
```

となる。

recursively expanded variables



プロフィール



id:minus9d

プログラミングや画像処理・機械学習の練習帳です。C/C++, Pythonが好きです。

読者になる

21

検索

ブログ内検索

最新記事

[ピタゴラス数を無限に生成する](#)

[PyInstallerでPythonスクリプトをexe化](#)

recursively expanded variablesは、コロンのイコールで定義される変数。実際に値が必要になった時に初めて定義式の右辺の変数が展開される。なので、recursively expanded variablesは、呼ばれるたびに異なる値となる可能性がある。

今度は下のMakefileを実行する。差分はコロンがなくなったことだけである。

```
CC = g++
MAKE_DEPEND = $(CC) -M

all:

    @printf "$(MAKE_DEPEND)"

CC = gcc
```

\$(MAKE_DEPEND)変数は、それが必要になったときにしか決定されない。\$(MAKE_DEPEND)が必要になるのはmake allを呼んだ時の@printfの中だが、その時点では\$(CC)はgccに上書きされているので、出力は

```
gcc -M
```

と、先ほどと異なる結果になる。

特に理由がなければ、=ではなく:=を使うほうが、意図しない動作を含みにくくて良いように思う。

値がセットされていないときのみ変数に値を代入

?= を使う。

特定のターゲットにのみ特別な変数を使う

あるターゲットをビルドするときだけ、あるフラグをONにしたい、といったときに以下の書き方ができる。

```
gui.o: CPPFLAGS += -DUSE_NEW_MALLOC=1
gui.o: gui.h
```

@と-の意味

@はコマンド行の実行を表示しない、という意味。
-はコマンド行の実行が失敗しても無視する、という意味。

例:

```
clean:

    -@rm *.o
```

暗黙のルールが使えない!?

以下のMakefileではルールが空である。

```
sample: sample.o sub1.o sub2.o
```

しかし、暗黙のルールが適用されて、以下のように実行ファイルsampleが生成される。

```
cc      -c -o sample.o sample.c
cc      -c -o sub1.o sub1.c
```

[PythonスクリプトをWindowsのexeにする方法 \(調査中\)](#)

[Python3では変数名に日本語が使える](#)

[matplotlibで軸の値が小数になったりオフセット表現になったりするのを止める方法](#)

人気記事

B エントリー
pythonでcsvを読む方法 - 標準ライブラリ, pandas, numpy - minus9d's diary 8users
matplotlibをオブジェクト指向スタイルで使う その2 - minus9d's diary 5users
EmacsのflymakeにてC++11の文法を自動チェックさせる - minus9d's diary 6users
matplotlibをオブジェクト指向スタイルで使う - minus9d's diary 6users
Pythonで、文字列の一部の文字を変更する - minus9d's diary 4users
閉形式(closed-form)とは - minus9d's diary 4users
OpenCVを使うならC++かPythonか? - minus9d's diary 3users
C++11のラムダ関数の簡単なまとめ - minus9d's diary 3users
C++11で数字 → 文字列は std::to_string()、文字列 → 数字は std::stoi()... 5users
はてなブログでtex記法を使うときのメモ - minus9d's diary 6users

月別アーカイブ

- ▶ 2016 (12)
- ▶ 2015 (46)
- ▼ 2014 (100)
 - 2014 / 12 (11)
 - 2014 / 11 (8)
 - 2014 / 10 (5)
 - 2014 / 9 (6)
 - 2014 / 8 (9)
 - 2014 / 7 (7)
 - 2014 / 6 (7)
 - 2014 / 5 (8)
 - 2014 / 4 (8)
 - 2014 / 3 (11)
 - 2014 / 2 (15)
 - 2014 / 1 (5)
- ▶ 2013 (75)
- ▶ 2012 (38)
- ▶ 2011 (23)
- ▶ 2010 (14)

```
cc      -c -o sub2.o sub2.c
cc      sample.o sub1.o sub2.o      -o sample
```

一方、以下のMakefileでは、暗黙のルールが適用されない。違いは何だろうか？

```
my_sample: sample.o sub1.o sub2.o
```

今回の場合、暗黙のルールが適用されるには、ターゲットの名前が、コロンの右側に羅列されたオブジェクトファイルのどれかと同じであることが必要なのであった。なので、最初の例では暗黙のルールが適用されたが、2番目の例では適用されなかった。

minus9d 2年前

★+

2

2

list

0

ブックマーク

シェア

ツイート

G+1



▶ 2009 (6)

カテゴリー

math (25)

programming contest (3)

python (52)

emacs (8)

english (5)

C (1)

Windows (12)

linux (23)

C++ (59)

zsh (5)

android (4)

google (5)

cygwin (2)

hatena (4)

machine learning (4)

雑文 (9)

others (7)

visual studio (11)

git (5)

opencv (16)

mac (9)

octave (5)

tex (3)

cv (2)

matlab (2)

gadget (1)

network (4)

unicode (1)

programming (10)

software (3)

gmail (1)

ubuntu (1)

twitter (1)

book (2)

powershell (2)
google code jam (1)
study (2)
google app engine (1)
voa (5)
web (1)
excel (2)
iPhone (1)
subversion (1)
eclipse (1)

最近のコメント

-  id:minus9d [matplotlibで軸の値が小数になっ...](#) (135日前)
-
-  id:kochory [matplotlibで軸の値が小数になっ...](#) (135日前)
-
-  id:minus9d [matplotlibで軸の値が小数になっ...](#) (136日前)
-
-  id:kochory [matplotlibで軸の値が小数になっ...](#) (136日前)
-
-  id:minus9d [C++11のtupleをvectorに突...](#) (1年前)
-

Amazon.co.jpアソシエイト