

5.3折5.9折4.5折5.6折5.9折3.6折

^

回頁頂 (gnu-make/variable/#main=content)

[GNU Make] Makefile 教學：如何在 Makefile 中設置變數



在前文的例子中，我們將所有的指令都寫死在 **Makefile** 中，這樣的做法雖然直觀，但不一定是最方便的做法。例如，筆者所用的某個雲端環境有 **GCC-4.8**、**GCC-4.9**、**Clang** 三套 C 編譯器，如果我們想在編譯時更換編譯器，按照先前的思維，勢必要頻繁修改 **Makefile** 或是撰寫三套 **Makefile**，讀者應該可以察覺到這不是很好的解決方法。

比較好的方法是將共通的部分以變數 (variable) 將 **Makefile** 參數化 (parameterization)，之後只要修改變數即可。我們將先前的例子參數化後改寫如下：

贊助商連結

▶ ×

...

5.9折

5.3折

5.2折

5.9折

贊助商連結

▶ ×

...

5.3折

5.9折

4.5折

5.3折

5.3折

5.9折

4.5折

5.6折

5.9折

3.6折

^



```
all: dynamic
回首頁 (/gnu-make/variable/#main-

dynamic:
回首頁 (/) $(CC) $(CFLAGS_RELEASE) -fPIC -c deque_int.c
$(CC) $(CFLAGS_RELEASE) -shared -o libalgodeque.so deque_int.o

$(TEST_PROG): $(OBJS)
$(CC) $(CFLAGS_DEBUG) -o $(TEST_PROG) $(OBJS)

test_deque_int.o:
$(CC) $(CFLAGS_DEBUG) -c test_deque_int.c

deque_int.o:
$(CC) $(CFLAGS_DEBUG) -c deque_int.c

clean:
$(RM) $(TEST_PROG) *.o *.so *.a
```

在 Makefile 中，會將變數設為全大寫，可和一般的系統指令區別；引用變數的格式是 \$(...)，將變數名稱塞入其中即可。

贊助商連結

贊助商連結

5.9折

5.3折

5.2折

5.9折

...
dynamic 任務中，我們使用了兩個參數，一個是 CC，一個是 CFLAGS_RELEASE，前者表示 C 編譯器 (C compiler)，後者表示編譯器所用的參數：

```
CC=gcc
CFLAGS_RELEASE=-Wall -Wextra -O2 -std=c99

dynamic:
$(CC) $(CFLAGS_RELEASE) -fPIC -c deque_int.c
$(CC) $(CFLAGS_RELEASE) -shared -o libalgodeque.so deque_int.o
```

告我們想用 Clang 編譯專案，只要將 CC 的值設為 clang 即可。由於 Clang 在參數上刻意和 GCC 保持相容，有時候 GCC 的參數可以原封不動地套用在 clang 上，在本例中剛好也可行。

接著，我們來看 test 任務的部分：

5.3折

5.9折

4.5折

5.3折

```
test: $(TEST_PROG)
    ./$(TEST_PROG)
回頁頂 (gnu-make/variable/#main-
echo $$?
content)
$(TEST_PROG): $(OBJS)
回頁頂 ()
    $(CC) $(CFLAGS_DEBUG) -o $(TEST_PROG) $(OBJS)
```

```
-c test_deque_int.c
```

```
-c deque_int.c
```

稱和相依性也可以設置為變數。

最後來看 *clean* 任務的部分：

```
clean:
    $(RM) $(TEST_PROG) *.o *.so *.a
```


細心的讀者可發現在本例的 **Makefile** 中，我們沒有設置 *RM* 變數，但專案卻可正常運作。這是因為 **make** 中有所謂的內隱變數 (implicit variable)，也就是有預設值的變數。在 **GNU Make** 中，*RM* 的預設值是 `rm -f`，剛好是強制刪除檔案的指令。

如果有使用 Windows 的 cmd 環境的讀者應該知道 cmd 環境中沒有 rm 指令，相對應的指令是 del。這是由於 GNU Make 是 Unix 文化的產物，自然會以類 Unix 系統的常見情境做為預設值。這不代表在 Windows 上無法使用 make 管理專案，只是要略為修改 Makefile 的寫法。

即使我們僅僅用了一小部分 **Makefile** 的語法，這個版本的 **Makefile** 的通用性比先前的版本更好一些了。學 **Makefile** 就像學程式設計般，並不是學完全部的特性才開始撰寫 **Makefile**，而是在一邊學習的過程中就可以開始寫簡單的 **Makefile**，隨著我們技巧增加，再逐步修改 **Makefile** 即可。

三、本文

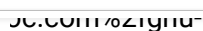
cebook (<https://www.facebook.com/sharer.php?>

[https%3a%2f%2fopensourcedoc.com%2fgnu-make%2fvariable%2f](https://3a%2f%2fopensourcedoc.com%2fgnu-make%2fvariable%2f)  Twitter

os://twitter.com/intent/tweet?url=https%3a%2f%2fopensourcedoc.com%2fgnu-
e%2fvariable%2f&text=%5bGNU%20Make%5d%20Makefile%20%e6%95%99%e5%
gkedIn (https://www.linkedin.com/shareArticle?

```
i=true&url=https%3a%2f%2fopensource.com%2fgnu-  
%2fvariable%2f&title=%5bGNU%20Make%5d%20Makefile%20%e6%95%99%e5%  
%e3%80%81GC&source=%e9%96%8b%e6%ba%90%e6%8a%80%e8%a1%93%e6%
```

3.6折



 Gmail (<https://mail.google.com/mail/?>



(<https://twitter.com/openSourceDok>)

0 Responses



Sad

贊助商連結



姓名

Copyright (c) 2014-2022 Michelle Chen. All Rights Reserved.