如何以 MinGW 和 MSYS2 建置 C 和 C++ 開發環境

Michael Chen 的技術文件 (https://michaelchen.tech/)
／ [Windows] 程式設計教學 (https://michaelchen.tech/windows-programming/)
／ 如何以 MinGW 和 MSYS2 建置 C 和 C++ 開發環境

請開啟瀏覽器的 JavaScript，以獲得最佳使用者體驗

本文最後修改日期為 MAY 11, 2021

本文共 1847 個字，閱讀時間約 5 分鐘

前言

在本文中，我們會用 MinGW + MSYS 建置 C 或 C++ 編譯環境；MinGW 是 GCC 在 Windows 上的移植品，而 MSYS 是小型的 POSIX 模擬環境，用來編譯軟體。

雖然 MinGW + MSYS 使用起來不若 Visual C++ 這種包山包海的 IDE 那麼方便，需要自行搭配其他的 IDE~~，但 MinGW 對 C 語言標準的支援比較進步，如果需要使用 C 語言新的特性的話，是值得考慮的替代性選擇~~。使用 MinGW 搭配 Code::Blocks 或 KDevelop 等 IDE，其實使用起來不會那麼不便。

原本的 MinGW 較不親民，我們會採用改良過的 MSYS2。到 MSYS2 官網 (https://www.msys2.org/) 下載安裝程式即可。~~根據 CPU 的位元數下載相對應的 MSYS2 安裝程式，按照指示安裝即可。在使用 64 位元的 Windows 系統時，不需刻意選 32 位元的 MSYS2 安裝程式，在 64 位元的 MSYS2 中即同時包含 32 和 64 位元的套件。~~現在 MSYS2 已經移除 32 位元的支援，只支援 64 位元的 Windows 7 以上系統。

MSYS2 分為五種子系統

使用 MSYS2 時，實際上我們有五種終端機環境：

- MinGW 32 bit：原生的 Windows 終端機環境，使用 32 bit 的 MinGW，搭配 msvcrt，用於編譯 32 位元的 Windows 原生軟體
- MinGW 64 bit：原生的 Windows 終端機環境，使用 64 bit 的 MinGW，搭配 msvcrt，用於編譯 64 位元的 Windows 原生軟體
- UCRT 64 bit：原生的 Windows 終端機環境，使用 64 bit 的 MinGW，搭配 ucrt，用於編譯 64 位元的 Windows 10 原生軟體

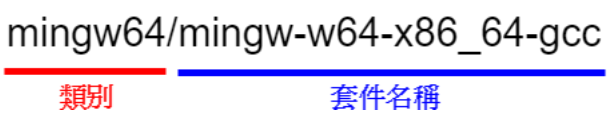
- Clang 64 bit：原生的 Windows 終端機環境，使用 64 bit 的 Clang，搭配 ucrt，用於編譯 64 位元的 Windows 10 原生軟體
- MSYS：特殊的 POSIX 子系統，僅用於編譯和安裝套件

我們要安裝套件時，會使用 MSYS 終端機環境，而要編譯 Windows 原生軟體時，就會回到其他終端機環境。現在 MSYS2 終端機環境的選擇比先前多了。不用感到慌亂，同一時間只會用到一種終端機環境。隨自己的需求來選擇即可。

MSYS2 在不同子環境下，可用的套件相異。msys2 的套件有五種字首：

- `mingw-w64-i686`：用於 MinGW 32 bit 終端機環境中
- `mingw-w64-x86_64`：用於 MinGW 64 bit 終端機環境中
- `mingw-w64-ucrt-x86_64`：用於 UCRT 64 bit 終端機環境中
- `mingw-w64-clang-x86_64`：用於 Clang 64 bit 終端機環境中
- 無字首：用於 MSYS 環境中

我們以 GCC 為例：



前面的 `mingw64` 代表這個套件屬於 MinGW 64 bit 環境，而後面一長串 `mingw-w64-x86_64-gcc` 則是實際的套件名稱。我們在終端機要安裝該套件時，需輸入套件名稱的部分。套件名稱有其邏輯在，但一般使用者不需在意這些細節。

在大部分的情形下，我們會視需求在各個原生 Windows 終端機環境中擇一安裝，至於 MSYS 套件則很少使用。因 MinGW 終端機環境可編譯出原生 Windows 應用程式，而 MSYS 終端機編譯出來的僅能在這個子系統中使用。

從命令提示字元開啟 MSYS2 終端機環境

以下指令會開啟 MinGW 64 bit 環境：

```
> c:\msys64\msys2_shell.cmd -mingw64
```

實際上，啟動選單裡的各個環境就是以此指令來呼叫。

第一次使用 MSYS2

現在假定我們剛裝好 MSYS2，要先更新 MSYS2 系統到最新狀態。開啟一個 MSYS 視窗，更新系統的核心組件：

```
$ pacman -Syu
```

更新完成後，關閉再重開 MSYS 視窗，更新其他部分：

```
$ pacman -Sy
```

用以下指令安裝 C 和 C++ 編譯器以及其他常見的開發工具：

```
$ pacman -S mingw-w64-x86_64-toolchain
```

安裝好開發工具後，可自行將相對應的路徑加入 PATH 變數，以 64 位元工具來說，將 C:\msys64\mingw64\bin 加入即可。如果不想影響系統的 PATH 環境變數，也可以使用 MinGW 終端機即可。

假定我們有修改環境變數。開啟 Windows 終端機，檢查 GCC 版本：

```
C:\> gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=(省略部分內容)
Target: x86_64-w64-mingw32
Configured with: (省略部分內容)
Thread model: posix
gcc version 7.2.0 (Rev1, Built by MSYS2 project)
```

如果出現某個 GCC 版本，代表 C 和 C++ 編譯環境建置成功。

安裝 MinGW 套件，以 GTK+ 為例

除了提供 C 和 C++ 編譯環境外，msys2 還將一些常見的 C 或 C++ 函式庫包裝成套件，簡化編譯套件的過程。在這裡，我們以 GTK+3 為例，說明如何使用 MSYS2 套件。

開啟 MSYS2 視窗，下載 64 位元版本的 GTK+3 套件：

```
$ pacman -S mingw-w64-x86_64-gtk3
```

在這裡，我們用 Go 社群的 gotk3 套件來展示使用套件的過程。請讀者先自行安裝 Go，再下載 gotk3 套件：

```
C:\> go get github.com/gotk3/gotk3/gtk
```

在這裡，以一個小型的視窗程式來試我們裝好的套件：

```
package main

import (
    "github.com/gotk3/gotk3/gtk"
    "log"
)

func main() {
    // Initialize GTK without parsing any command line arguments.
    gtk.Init(nil)

    // Create a new toplevel window, set its title, and connect it to the
    // "destroy" signal to exit the GTK main loop when it is destroyed.
    win, err := gtk.WindowNew(gtk.WINDOW_TOPLEVEL)
    if err != nil {
        log.Fatal("Unable to create window:", err)
    }
    win.SetTitle("Simple Example")
    win.Connect("destroy", func() {
        gtk.MainQuit()
    })

    // Create a new label widget to show in the window.
    l, err := gtk.LabelNew("Hello, gotk3!")
    if err != nil {
        log.Fatal("Unable to create label:", err)
    }

    // Add the label to the window.
    win.Add(l)

    // Set the default window size.
    win.SetDefaultSize(800, 600)

    // Recursively show all widgets contained in this window.
    win.ShowAll()

    // Begin executing the GTK main loop.  This blocks until
    // gtk.MainQuit() is run.
    gtk.Main()
}
```

編譯此程式後執行：

```
C:\> go build demo.go
C:\> .\demo.exe
```

要注意的是，如果在電腦上裝多套 C 和 C++ 編譯環境及一些函式庫，像是 Strawberry Perl 內附的 GCC，要把 msys2 所在的路徑向前移，才會正確呼叫到相對應的函式庫。

結語

在本文中，我們介紹了 MSYS2 (MinGW + MSYS 新版本) 的使用方式，這套軟體可以取代 Visual Studio，用來撰寫 C 或 C++ 應用程式。