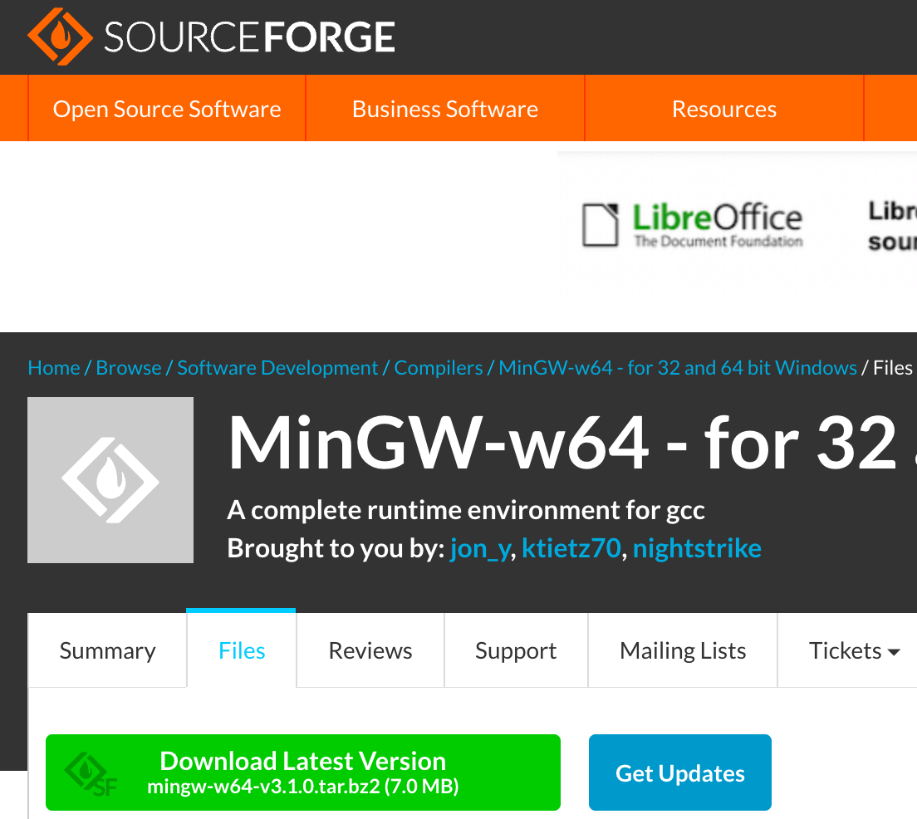
**python\_c\_c2 call dll**

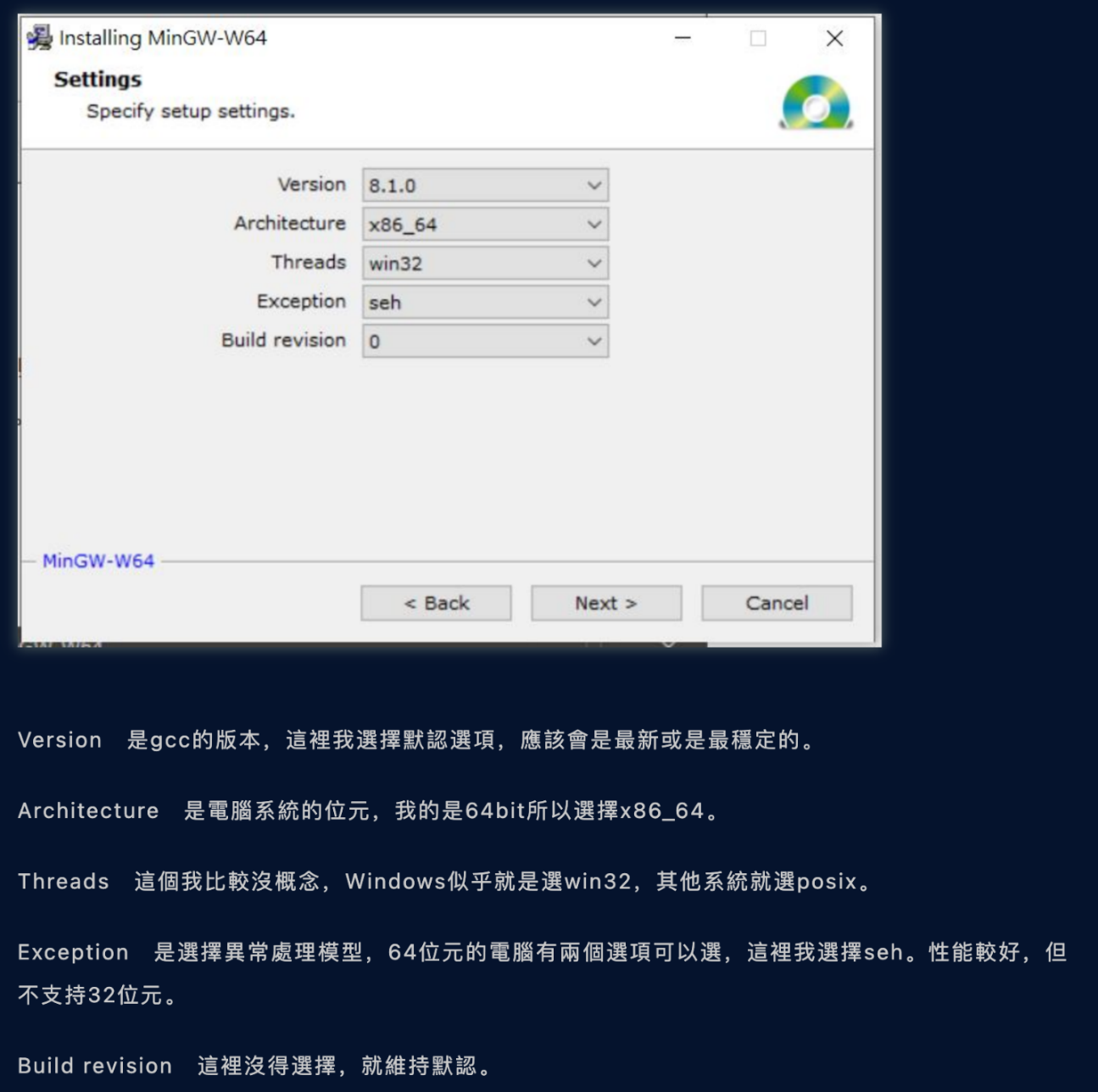
1. MinGW-w64 GCC install
2. Python call dll for c
3. Python call dll for c2
4. C call dll
5. C2 call dll
6. continue
7. MinGW-w64 GCC install

MinGW-w64是要在Windows環境下使用gcc的方式，Windows版的gcc。

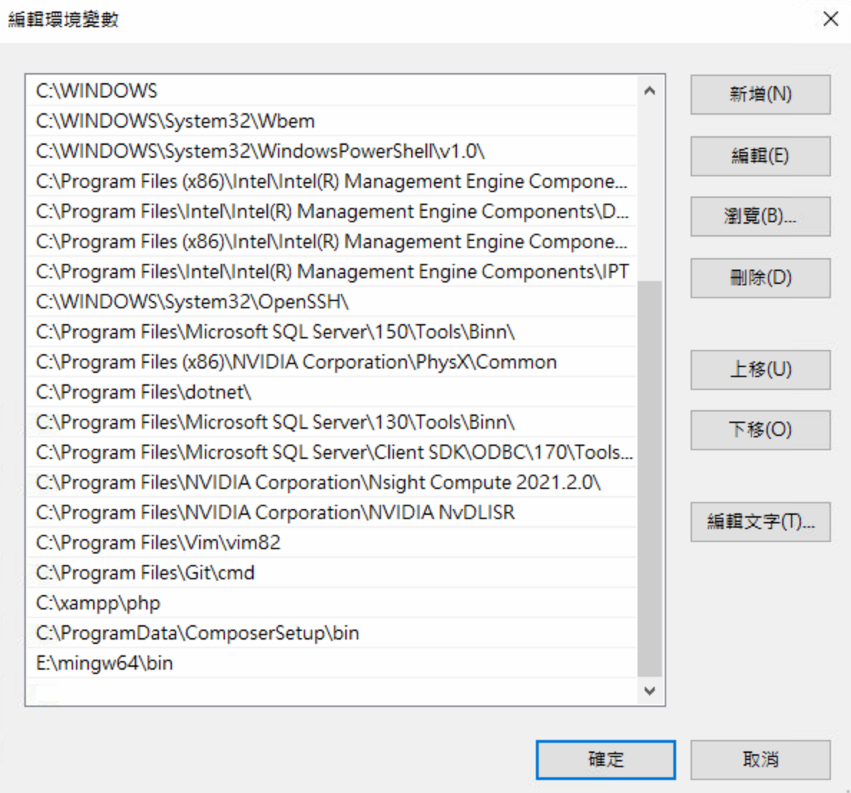
* 1. MinGW-W64 GCC安裝:
     1. Address: <https://sourceforge.net/projects/mingw-w64/>
     2. 選擇的檔案，並且下載。



* + 1. 選擇x86\_64-win32-seh的意思，下面畫面是網路安裝版出現的畫面，若是下載的版本則是像這樣的檔案名稱 “x86\_64-win32-seh”。



* 1. 下載後的檔案可以放在C、D、E… …槽都可，然後設定環境變數路徑。



* 1. 在command line執行gcc -v 或是 g++ -v，若成功將出現以下畫面。



1. python call dll for c
   1. edit TEST01.c file

|  |  |  |
| --- | --- | --- |
| **#include <stdio.h>** | **1** |  |
| **//#include “TEST01.h”** | **2** |  |
|  | **3** |  |
| **long long calc\_factorial(long long \_num) {** | **4** |  |
| **long long i;** | **5** |  |
| **long long \_result = 1;** | **6** |  |
|  | **7** |  |
| **for (i = 1; i <= \_num; i++) {** | **8** |  |
| **\_result = \_result \* i;** | **9** |  |
| **}** | **10** |  |
|  | **11** |  |
| **return \_result;** | **13** |  |
|  | **13** |  |
|  | **14** |  |

* 1. edit TEST02.c file

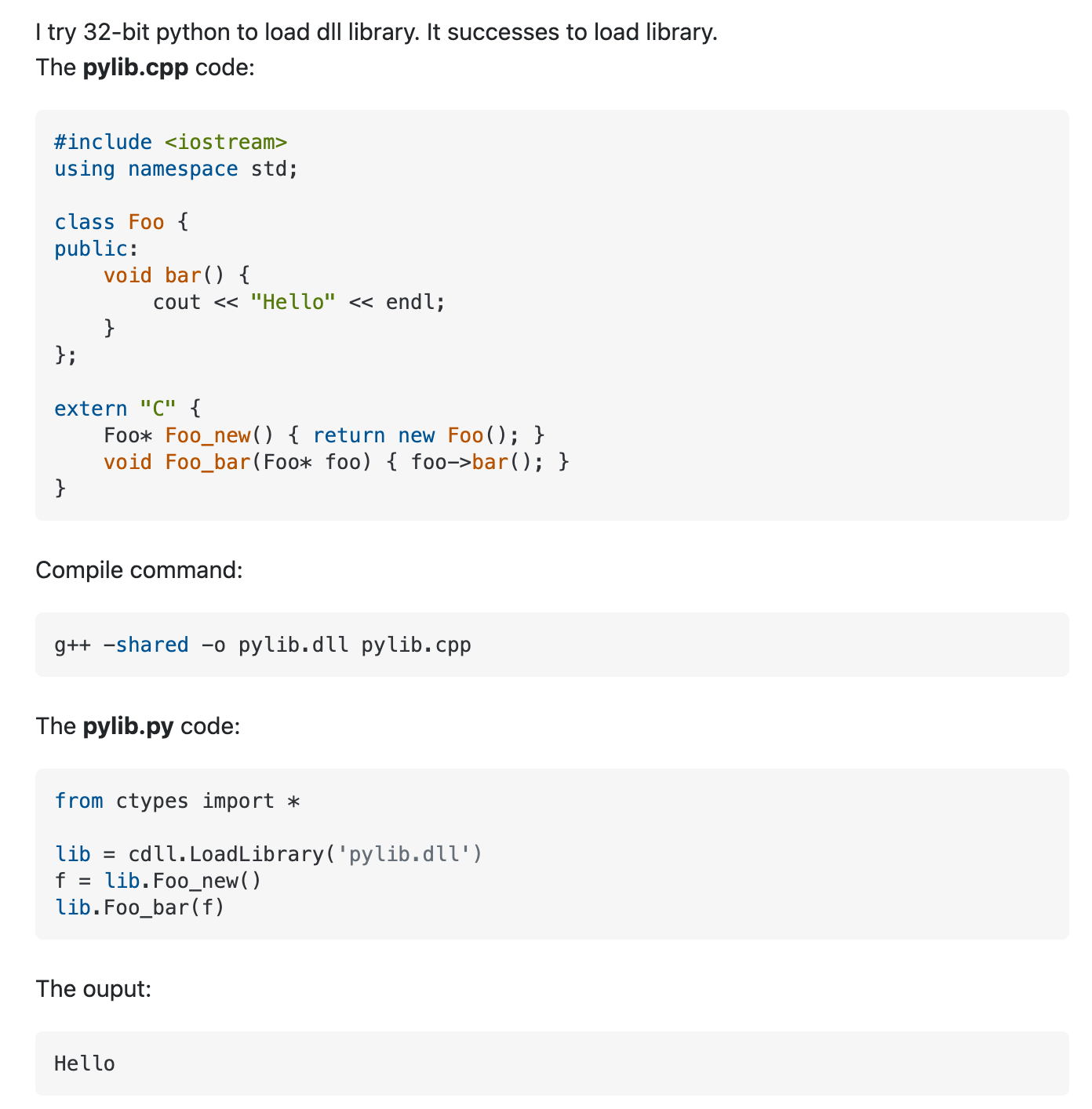
|  |  |  |
| --- | --- | --- |
| **#include <stdio.h>** | **1** |  |
| **#include “TEST01.h”** | **2** |  |
|  | **3** |  |
| **void main(void) {** | **4** |  |
| **int result;** | **5** |  |
|  | **6** |  |
| **printf(“ this is test02.c\n”);** | **7** |  |
| **result = calc\_factorial(10);** | **8** |  |
| **printf(“>>> show factorial: %d\n”, result);** | **9** |  |
| **}** | **10** |  |
|  | **11** |  |

* 1. edit call\_dll.py 腳本

|  |  |  |
| --- | --- | --- |
| **import ctypes** | **1** |  |
| **from ctypes import \*** | **2** |  |
| **import os** | **3** |  |
| **import time** | **4** |  |
| **import datetime** | **5** |  |
|  | **6** |  |
| **CUR\_PATH = os.getcwd()** | **7** |  |
| **DLL\_PATH = os.path.join(CUR\_PATH, “TEST01.dll”)** | **8** |  |
|  | **9** |  |
| **if \_\_name\_\_ == “\_\_main\_\_”:** | **10** |  |
| **print(“>>> center call\_dll.py”)** | **11** |  |
| **print(“>>> show DLL PATH: “, DLL\_PATH)** | **13** |  |
|  | **13** |  |
| **my\_dll = ctypes.cdll.LoadLibrary(DLL\_PATH)** | **14** |  |
| **my\_dll.calc\_factorial.argtypes = [c\_longlong]** | **15** | 設定 DLL 檔案中 dist 函數的參數資料型態 |
| **my\_dll.calc\_factorial.restype = c\_longlong** | **16** | 設定 DLL 檔案中 dist 函數的傳回值資料型態 |
| **result = my\_dll.calc\_factorial(c\_longlong(15))** | **17** |  |
|  | **18** |  |
| **print(“>>> show result of factorial: “, result)** | **19** |  |
|  | **20** |  |

* 1. 呼叫dll方法1: 製作TEST01.c成為TEST01.dll
     1. Step1: 執行gcc -shared -o ./TEST01.dll ./TEST01.c。
     2. Step2: 執行!python call\_dll.py。
  2. 呼叫dll方法2: 製作TEST01.o成為TEST01.dll
     1. Step1: 執行gcc -c ./TEST01.c，產生TEST01.o的檔案。
     2. Step2: 執行gcc -shared -o ./TEST01.dll ./TEST01.o，產生TEST01.dll檔案。
     3. Step3: 執行!python call\_dll.py。
  3. argtypes的參數資料型態:
     1. c\_int
     2. c\_double
     3. c\_longlong

1. python call dll for c2
   1. 需要將程式改成如下圖的狀態。



1. c call dll
   1. Step1: 執行gcc -shared -o ./TEST01.dll ./TEST01.c，產生TEST01.dll的檔案。
   2. Step2: 執行gcc -o app ./TEST01.dll ./TEST02.c，產生app.exe執行檔。
2. c2 call dll: 要將extern “C” { }，的包覆拿開。
   1. Step1: 執行g++ -shared -o ./TEST01.dll ./TEST01.cpp，產生TEST01.dll的檔案。
   2. Step2: 執行g++ -o app ./TEST01.dll ./TEST02.cpp，產生app.exe執行檔。
3. continue