

11-9 節(多位元組字串)補充說明

由於不同作業系統對於中文編碼處理上的差異，原第 11 章 11-9 節中的 Example 11-15 與 Example 11-16 範例程式僅能在 Linux 作業系統上執行，並無法在 Windows 與 macOS 上正確地執行；甚至部份使用 Linux 的讀者，由於繁體中文環境的設定不同，也可能無法正確地執行這些範例。為了幫助讀者能夠正確地處理多位元組字串，筆者在此以繁體中文最普及的 zh_TW.UTF-8 編碼為例，為讀者清楚說明如何才能在 Linux、Windows 與 macOS 作業系統上，正確地處理多位元組字串輸入輸出的補充說明。請有需要的讀者參閱相關的小節。

Linux 作業系統

使用 Linux 作業系統的讀者，只要已安裝有 zh_TW.UTF-8 的繁體中文環境，原則上就可以依照本書 11-9 節的內容完成相關範例的執行。若是不確定自己所使用的系統是否已正確安裝有 zh_TW.UTF-8 繁體中文環境，讀者可以在終端機裡使用 `locale -a | grep zh_TW.UTF-8` 指令進行檢查是否已安裝¹——此指令的執行結果若輸出一行 zh_TW.UTF-8，就代表已安裝；反之，若是沒有任何輸出結果，則代表未安裝。

針對尚未安裝 zh_TW.UTF-8 的讀者，請使用以下指令進行系統更新並安裝繁體中文的語言包：

```
sudo apt update
sudo apt install -y language-pack-zh-hant
```

完成安裝之後，請再使用以下的指令來設定使用 zh_TW.UTF-8 做為預設的中文編碼：

```
sudo locale-gen zh_TW.UTF-8
```

¹ 由於 Linux 作業系統有許多不同的發行版，在此筆者僅就較多人使用的 Ubuntu 發行版為例，為讀者說明相關的指令操作；使用其它發行版的讀者，若遇指令不同的情況，請自行修改使用其它指令。

上述的安裝與設定操作全部完成後，讀者可以再次使用 `locale-a|grep zh_TW.UTF-8` 指令查看是否已安裝及設定成功。若是一切順利的話，現在開始就可以在系統上正確地使用 C++ 語言完成繁體中文字串輸入輸出的程式設計了。

Windows 作業系統

大部份的 Windows 作業系統使用 UTF-16 做為預設的繁體中文編碼，讀者應該不需要自行安裝，但為了將原本在 Linux 系統裡執行的處理多位元組字串的 C++ 程式（例如 Example 11-15 與 Example 11-16）移植到 Windows 系統裡，其中用以設定程式所使用的字元編碼的敘述則必須加以變更。具體來說，在本書第 11 章的 Example 11-15 與 Example 11-16 範例程式，它們的第 6 至第 8 行都具有以下相同的內容：

```
ios_base::sync_with_stdio(false);
wcin.imbue(locale("zh_TW.UTF-8"));
wcout.imbue(locale("zh_TW.UTF-8"));
```

其作用就是在 Linux 系統上執行時，設定使用 zh_TW.UTF-8 的編碼，請將其加以加以移除，並改為使用以下的敘述來設定使用 UTF-16 做為標準輸出與輸入的編碼方式：

```
_setmode(_fileno(stdout), _O_U16TEXT);
_setmode(_fileno(stdin), _O_U16TEXT);
```

其中 `_O_U16TEXT` 是定義在 `fcntl.h` 標頭檔裡的常數，請記得將此標頭檔載入，如此一來，此程式在 Windows 系統裡執行時，就會正確地使用繁體中文的編碼了。為了便利讀者起見，筆者將移植到 Windows 系統裡執行的 Example 11-15 與 Example 11-16 進行上述的修改，並列示如下：

Windows →

Example 11-15：使用 `wchar_t` 處理繁體中文的輸入與輸出範例

Location:  examples/ch11

Filename: `wchar_t_win.cpp`

1	#include <iostream>
2	#include <fcntl.h>

```

3 using namespace std;
4
5 int main()
6 {
7     _setmode(_fileno(stdout), _O_U16TEXT);
8     _setmode(_fileno(stdin), _O_U16TEXT);
9
10    wchar_t otani[]=L"大谷翔平";
11    wchar_t baseball_1=L'棒', baseball_2=L'球';
12    wchar_t name[10];
13
14    wcout << L"請輸入你的名字:";
15    wcin >> name;
16
17    wcout << name << L"最欣賞的";
18    wcout << baseball_1 << baseball_2;
19    wcout << L"選手是";
20
21    for(int i=0;i<wcslen(otani); i++)
22        wcout << otani[i];
23
24    wcout << endl;
25    return 0;
26 }

```

此範例程式和原本的 Example 11-15 範例程式 `wchar_t.cpp` 的內容大致相同，但在第 2 行載入了 `fcntl.h` 標頭檔，並在第 7 行與第 8 行使用 `_setmode()` 函式設定了標準輸出（`stdout`）與標準輸入（`stdin`）皆使用 UTF-16 編碼，除此之外，後續的程式碼和原本的內容皆相同。

Windows →

Example 11-16：使用 `wstring` 處理繁體中文的輸入與輸出範例

Location: /examples/ch11

Filename: `wstring_win.cpp`

```

1 #include <iostream>
2 #include <fcntl.h>
3 using namespace std;

```

```

4
5 int main()
6 {
7     _setmode(_fileno(stdout), _O_U16TEXT);
8     _setmode(_fileno(stdin), _O_U16TEXT);
9
10    wstring otani=L"大谷翔平";
11    wchar_t baseball_1=L'棒', baseball_2=L'球';
12    wstring name;
13
14    wcout << L"請輸入你的名字:";
15    wcin >> name;
16
17    wcout << name << L"最欣賞的";
18    wcout << baseball_1 << baseball_2;
19    wcout << L"選手是";
20
21    for(int i=0;i<otani.length(); i++)
22        wcout << otani[i];
23
24    wcout << endl;
25    return 0;
26 }

```

如同上一個範例一樣，此程式同樣只有第 2 行載入 `fcntl.h` 標頭檔與第 7 行、第 8 行設定標準輸入與標準輸出使用 UTF-16 編碼，不同於 Linux 的版本以外，其餘部份皆相同。

macOS 作業系統

macOS 作業系統和 Linux 相同，都是使用 UTF-8 做為中文編碼，但其設定程式使用 UTF-8 的方式和 Linux 系統上有所不同。具體來說，在本書第 11 章的 Example 11-15 與 Example 11-16 範例程式，它們的第 6 至第 8 行都具有以下相同的內容：

```
ios_base::sync_with_stdio(false);
```

```
wcin.imbue(locale("zh_TW.UTF-8"));
wcout.imbue(locale("zh_TW.UTF-8"));
```

其作用就是在 Linux 系統上執行時，設定使用 zh_TW.UTF-8 的編碼，請將其加以移除，並改為使用以下的敘述來設定使用 `codecvt` 標頭檔中所定義的編碼轉換功能，產生一個 `codecvt_utf8` 類別的物件，並用以設定做為寬字元的輸入與輸出編碼（別忘了要載入 `codecvt` 標頭檔）：

```
locale utf8_locale(locale(), new codecvt_utf8<wchar_t>());
wcin.imbue(utf8_locale);
wcout.imbue(utf8_locale);
```

為了便利讀者起見，筆者將移植到 macOS 系統裡執行的 Example 11-15 與 Example 11-16 進行上述的修改，並列示如下：

macOS →

Example 11-15：使用 `wchar_t` 處理繁體中文的輸入與輸出範例

Location: /examples/ch11

Filename: `wchar_t_macos.cpp`

```
1 #include <iostream>
2 #include <codecvt>
3 using namespace std;
4
5 int main()
6 {
7     locale utf8_locale(locale(),
8                          new codecvt_utf8<wchar_t>());
9     wcin.imbue(utf8_locale);
10    wcout.imbue(utf8_locale);
11
12    wchar_t otani[] = L"大谷翔平";
13    wchar_t baseball_1 = L'棒', baseball_2 = L'球';
14    wchar_t name[10];
15
16    wcout << L"請輸入你的名字:";
17    wcin >> name;
```

```

18     wcout << name << L"最欣賞的";
19     wcout << baseball_1 << baseball_2;
20     wcout << L"選手是";
21
22     for(int i=0;i<wcslen(otani); i++)
23         wcout << otani[i];
24
25     wcout << endl;
26     return 0;
27 }

```

此範例程式和原本的 Example 11-15 範例程式 `wchar_t.cpp` 的內容大致相同，但在第 2 行載入了 `codecvt` 標頭檔，並在第 7 行至第 9 行設定 UTF-8 編碼，除此之外，後續的程式碼和原本的內容皆相同。

macOS →

Example 11-16: 使用 `wstring` 處理繁體中文的輸入與輸出範例

Location: /examples/ch11

Filename: `wstring_macos.cpp`

```

1 #include <iostream>
2 #include <codecvt>
3 using namespace std;
4
5 int main()
6 {
7     locale utf8_locale(locale(),
8                         new codecvt_utf8<wchar_t>());
9     wcin.imbue(utf8_locale);
10    wcout.imbue(utf8_locale);
11
12    wstring otani=L"大谷翔平";
13    wchar_t baseball_1=L'棒', baseball_2=L'球';
14    wstring name;
15
16    wcout << L"請輸入你的名字:";
17    wcin >> name;

```

```
18     wcout << name << L"最欣賞的";
19     wcout << baseball_1 << baseball_2;
20     wcout << L"選手是";
21
22     for(int i=0;i<otani.length(); i++)
23         wcout << otani[i];
24
25     wcout << endl;
26     return 0;
27 }
```

如同上一個範例一樣，此程式同樣只有第 2 行載入 `codecvt` 標頭檔與第 7 行至第 9 行設定使用 UTF-8 編碼，其餘部份皆相同。

