IT人 (/)

gcc 和 g++ 的聯絡和區別，使用 gcc 編譯 C++

wohu1104 發表於 2020-12-05

C++ (/topic/33.html)

GCC 編譯器已經為我們提供了呼叫它的介面，對於 C 語言或者 C++ 程式，可以通過執行 gcc 或者 g++ 指令來呼叫 GCC 編譯器。

實際使用中我們更習慣使用 gcc 指令編譯 C 語言程式，用 g++ 指令編譯 C++ 程式碼。需要強調的一點是，這並不是 gcc 和 g++ 的區別，gcc 指令也可以用來編譯 C++ 程式，同樣 g++ 指令也可以用於編譯 C 語言程式。

實際上，只要是 GCC 支援編譯的程式程式碼，都可以使用 gcc 命令完成編譯。可以這樣理解，gcc 是 GCC 編譯器的通用編譯指令，因為根據程式檔案的字尾名，gcc 指令可以自行判斷出當前程式所用程式語言的類別，比如：

- xxx.c ：預設以編譯 C 語言程式的方式編譯此檔案
- xxx.cpp ：預設以編譯 C++ 程式的方式編譯此檔案
- xxx.m ：預設以編譯 Objective-C 程式的方式編譯此檔案
- xxx.go ：預設以編譯 Go 語言程式的方式編譯此檔案

當然，gcc 指令也為使用者提供了 “手動指定代表編譯方式” 的介面，即使用 -x 選項：

- gcc -xc xxx 表示以編譯 C 語言程式碼的方式編譯 XXX 檔案
- gcc -xc++ xxx 則表示以編譯 C++ 程式碼的方式編譯 XXX 檔案

但如果使用 g++ 指令，則無論目標檔案的字尾名是什麼，該指令都一律按照編譯 C++ 程式碼的方式編譯該檔案。也就是說，

- 對於 .c 檔案來說，gcc 指令以 C 語言程式碼對待，g++ 指令會以 C++ 程式碼對待。
- 但對於 .cpp 檔案來說，gcc 和 g++ 都會以 C++ 程式碼的方式編譯。

C++ 標準對程式碼書寫規範的要求更加嚴格。

除此之外對於編譯執行 C++ 程式，使用 gcc 和 g++ 也是有區別的。要知道，很多 C++ 程式都會呼叫某些標準庫中現有的函式或者類物件，而單純的 gcc 命令是無法自動連結這些標準庫檔案的。舉個例子：

```
//test.cpp
#include <iostream>
#include <string>
using namespace std;

int main(){
    string str ="hello,world";
    cout << str << endl;
    return 0;
}
```

這是一段很簡單的 C++ 程式，其通過 <string> 標頭檔案提供的 string 字串類定義了一個字串物件，隨後使用 cout 輸出流物件將其輸出。對於這段 C++ 程式碼，如果我們使用 g++ 指令編譯，如下所示：

```
[wohu@ubuntu~]$ g++ test.cpp
[wohu@ubuntu~]$
```

可以看到，整個編譯過程沒有報任何錯誤。但如果使用 gcc 指令：

```
[wohu@ubuntu~]$ gcc test.cpp
/tmp/ccluIPFx.o: In function `main':
test.cpp:(.text+0x20): undefined reference to `std::allocator<char>::allocator()'
....
/tmp/ccluIPFx.o:(.eh_frame+0x13): undefined reference to `__gxx_personality_v0'
collect2: error: ld returned 1 exit status
```

其根本原因就在於，該程式中使用了標準庫 <iostream> 和 <string> 提供的類物件，而 gcc 預設是無法找到它們的。

如果想使用 gcc 指令來編譯執行 C++ 程式，需要在使用 gcc 指令時，手動為其新增 -lstdc++ 選項，表示 gcc 在編譯 C++ 程式時可以連結必要的 C++ 標準庫。如下示例：

```
[wohu@ubuntu~]$ gcc test.cpp -lstdc++
[wohu@ubuntu~]$
```

由此，test.cpp 就被成功的編譯了。

對於 gcc 和 g++ 指令，還有其它更多細節方面的區別，當前只需要知道：

- 對於 C 語言程式的編譯，我們應該使用 gcc 指令
- 編譯 C++ 程式則推薦使用 g++ 指令

這就足夠了。

相關文章

抽象類和介面的區別 (/97488.html)

2020-12-28

c++11-17 模板核心知識（十二）—— 模板的模板引數 Template Template Parameters (/564836.html)

2020-12-04 C++ (/topic/33.html)

【C++設計模式】組合模式 (/564978.html)

2020-12-04 C++ (/topic/33.html) 設計模式 (/topic/105.html)

CCF CSP201903-4訊息傳遞介面（c++100）(/565077.html)

2020-12-04 C++ (/topic/33.html)