Compiling Cpp

出自**Ubuntu**中文

C++ 編程中相關文件後綴

.a	靜態庫 (archive)
.C **.c** **.cc** **.cp** **.cpp** **.cxx** **.c++**	C++源代碼（需要編譯預處理）
.h	C或者C++源代碼頭文件

目錄

.ii	C++源代碼（不需編譯預處理）
.o	對象文件
.s	彙編語言代碼
.so	動態庫
<none>	標準C++系統頭文件

單個源文件生成可執行程序

下面是一個保存在文件 helloworld.cpp 中一個簡單的 C++ 程序的代碼：

```
/* helloworld.cpp */
#include <iostream>
int main(int argc,char *argv[])
{
    std::cout << "hello, world" << std::endl;
    return(0);
}
```

程序使用定義在頭文件 iostream 中的 cout，向標準輸出寫入一個簡單的字符串。該代碼可用以下命令編譯為可執行文件：

```
$ g++ helloworld.cpp
```

編譯器 g++ 通過檢查命令行中指定的文件的後綴名可識別其為 C++ 源代碼文件。編譯器默認的動作：編譯源代碼文件生成對象文件(object file)，鏈接對象文件和 libstdc++ 庫中的函數得到可執行程序。然後刪除對象文件。由於命令行中未指定可執行程序的文件名，編譯器採用默認的 a.out。程序可以這樣來運行：

```
$ ./a.out
hello, world
```

更普遍的做法是通過 -o 選項指定可執行程序的文件名。下面的命令將產生名為 helloworld 的可執行文件：

```
$ g++ helloworld.cpp -o helloworld
```

在命令行中輸入程序名可使之運行：

```
$ ./helloworld
hello, world
```

程序 g++ 是將 gcc 默認語言設為 C++ 的一個特殊的版本，鏈接時它自動使用 C++ 標準庫而不用 C 標準庫。通過遵循源碼的命名規範並指定對應庫的名字，用 gcc 來編譯鏈接 C++ 程序是可行的，如下例所示：

```
$ gcc helloworld.cpp -lstdc++ -o helloworld
```

選項 -l (ell) 通過添加前綴 lib 和後綴 .a 將跟隨它的名字變換為庫的名字 libstdc++.a。而後它

在標準庫路徑中查找該庫。gcc 的編譯過程和輸出文件與 g++ 是完全相同的。

在大多數系統中，GCC 安裝時會安裝一名為 c++ 的程序。如果被安裝，它和 g++ 是等同，如下例所示，用法也一致：

```
$ c++ helloworld.cpp -o helloworld
```

多個源文件生成可執行程序

如果多於一個的源碼文件在 g++ 命令中指定，它們都將被編譯並被鏈接成一個單一的可執行文件。下面是一個名為 speak.h 的頭文件；它包含一個僅含有一個函數的類的定義：

```
/* speak.h */
#include <iostream>
class Speak
{
    public:
        void sayHello(const char *);
};
```

下面列出的是文件 speak.cpp 的內容：包含 sayHello() 函數的函數體：

```
/* speak.cpp */
#include "speak.h"
void Speak::sayHello(const char *str)
{
    std::cout << "Hello " << str << "\n";
}
```

文件 hellospeak.cpp 內是一個使用 Speak 類的程序：

```
/* hellospeak.cpp */
#include "speak.h"
int main(int argc,char *argv[])
{
    Speak speak;
    speak.sayHello("world");
    return(0);
}
```

下面這條命令將上述兩個源碼文件編譯鏈接成一個單一的可執行程序：

```
$ g++ hellospeak.cpp speak.cpp -o hellospeak
```

PS：這裏說一下為什麼在命令中沒有提到“speak.h“該文件（原因是：在“speak.cpp“中包含有”#include"speak.h"“這句代碼，它的意思是搜索系統頭文件目錄之前將先在當前目錄中搜索文件“speak.h“。而”speak.h“正在該目錄中，不用再在命令中指定了）。

源文件生成對象文件

選項 -c 用來告訴編譯器編譯源代碼但不要執行鏈接，輸出結果為對象文件。文件默認名與源碼文件名相同，只是將其後綴變為 .o。例如，下面的命令將編譯源碼文件 hellospeak.cpp 並生成對象文件 hellospeak.o：

```
$ g++ -c hellospeak.cpp
```

命令 g++ 也能識別 .o 文件並將其作為輸入文件傳遞給鏈接器。下列命令將編譯源碼文件為對象文件並將其鏈接成單一的可執行程序：

```
$ g++ -c hellospeak.cpp
$ g++ -c speak.cpp
$ g++ hellospeak.o speak.o -o hellospeak
```

選項 -o 不僅僅能用來命名可執行文件。它也用來命名編譯器輸出的其他文件。例如：除了中間的對象文件有不同的名字外，下列命令生將生成和上面完全相同的可執行文件：

```
$ g++ -c hellospeak.cpp -o hspk1.o
$ g++ -c speak.cpp -o hspk2.o
$ g++ hspk1.o hspk2.o -o hellospeak
```

編譯預處理

選項 -E 使 g++ 將源代碼用編譯預處理器處理后不再執行其他動作。下面的命令預處理源碼文件 helloworld.cpp 並將結果顯示在標準輸出中：

```
$ g++ -E helloworld.cpp
```

本文前面所列出的 helloworld.cpp 的源代碼，僅僅有六行，而且該程序除了顯示一行文字外什麼都不做，但是，預處理后的版本將超過 1200 行。這主要是因為頭文件 iostream 被包含進來，而且它又包含了其他的頭文件，除此之外，還有若干個處理輸入和輸出的類的定義。

預處理過的文件的 GCC 後綴為 .ii，它可以通過 -o 選項來生成，例如：

```
$ gcc -E helloworld.cpp -o helloworld.ii
```

生成彙編代碼

選項 -S 指示編譯器將程序編譯成彙編語言，輸出彙編語言代碼而後結束。下面的命令將由 C++ 源碼文件生成彙編語言文件 helloworld.s：

```
$ g++ -S helloworld.cpp
```

生成的彙編語言依賴於編譯器的目標平台。

創建靜態庫

靜態庫是編譯器生成的一系列對象文件的集合。鏈接一個程序時用庫中的對象文件還是目錄中的對象文件都是一樣的。庫中的成員包括普通函數，類定義，類的對象實例等等。靜態庫的另一個名字叫歸檔文件(archive)，管理這種歸檔文件的工具叫 ar 。

在下面的例子中，我們先創建兩個對象模塊，然後用其生成靜態庫。

頭文件 say.h 包含函數 sayHello() 的原型和類 Say 的定義：

```
/* say.h */
#include <iostream>
void sayhello(void);
class Say {
    private:
        char *string;
    public:
        Say(char *str)
        {
            string = str;
        }
        void sayThis(const char *str)
        {
            std::cout << str << " from a static library\n";
        }
        void sayString(void);
};
```

下面是文件 say.cpp 是我們要加入到靜態庫中的兩個對象文件之一的源碼。它包含 Say 類中 sayString() 函數的定義體；類 Say 的一個實例 librarysay 的聲明也包含在內：

```
/* say.cpp */
#include "say.h"
void Say::sayString()
{
    std::cout << string << "\n";
}

Say librarysay("Library instance of Say");
```

源碼文件 syshello.cpp 是我們要加入到靜態庫中的第二個對象文件的源碼。它包含函數 sayhello() 的定義：

```
/* sayhello.cpp */
#include "say.h"
void sayhello()
{
    std::cout << "hello from a static library\n";
}
```

下面的命令序列將源碼文件編譯成對象文件，命令 ar 將其存進庫中：

```
$ g++ -c sayhello.cpp
$ g++ -c say.cpp
$ ar -r libsay.a sayhello.o say.o
```

程序 ar 配合參數 -r 創建一個新庫 libsay.a 並將命令行中列出的對象文件插入。採用這種方法，如果庫不存在的話，參數 -r 將創建一個新的庫，而如果庫存在的話，將用新的模塊替換原來的模塊。

下面是主程序 saymain.cpp，它調用庫 libsay.a 中的代碼：

```
/* saymain.cpp */
#include "say.h"
int main(int argc,char *argv[])
{
    extern Say librarysay;
    Say localsay = Say("Local instance of Say");
    sayhello();
    librarysay.sayThis("howdy");
    librarysay.sayString();
    localsay.sayString();
    return(0);
}
```

該程序可以下面的命令來編譯和鏈接：

```
$ g++ saymain.cpp libsay.a -o saymain
```

程序運行時，產生以下輸出：

```
hello from a static library
howdy from a static library
Library instance of Say
Local instance of Say
```

其他參考

- GCC新手入門
- C/C++ IDE簡介
- 用GDB調試程序
- Gtk與Qt編譯環境安裝與配置
- 跟我一起寫Makefile
- C編譯初步
- C++編譯初步
- Fortran編譯初步
- C和C++混合編譯初步
- C和Fortran混合編譯初步