Compiling C

出自**Ubuntu**中文

C 編程中相關文件後綴

.a 靜態庫 (archive)
.c C源代碼（需要編譯預處理）
.h C源代碼頭文件
.i 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

單個源文件生成可執行程序

目錄

下面是一個簡單的“hello, ubuntu”程序的源代碼：

```
/* helloubuntu.c */
#include <stdio.h>
int main(int argc,char *argv[])
{
          printf("hello, ubuntu\n");

          return 0;
}
```

最簡單直接的編譯該代碼為可執行程序的方法是，將該代碼保存為文件 helloubuntu.c，並執行以下命令：

```
$ gcc -Wall helloubuntu.c
```

編譯器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。GCC 默認的動作：編譯源代碼文件生成對象文件(object file)，鏈接對象文件得到可執行程序，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。在命令行中輸入程序名可使其執行並顯示結果：

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

選項 -o 用來指定所生成的可執行程序的文件名。下面的命令生成名為 helloubuntu 的可執行程序：

```
$ gcc -Wall helloubuntu.c -o helloubuntu
```

在命令行中輸入程序名將使其執行，如下：

```
$ ./helloubuntu
hello, ubuntu
```

注意如果有用到**math.h**庫等非**gcc**默認調用的標準庫，請使用 -lm參數

源文件生成對象文件

選項 -c 指示 GCC 編譯源代碼文件，但將對象文件保留在磁盤中並跳過鏈接對象文件生成可執行文件這一步。在這種情況下，默認的輸出文件的文件名同源代碼文件名一致，只不過後綴換為 .o 。例如：下面的命令將生成名為 helloubuntu.o 的對象文件：

```
$ gcc -c -Wall helloubuntu.c
```

選項 -o 可用來指定生成的對象文件的文件名。以下命令將產生名為kubuntu.o的對象文件：

```
$ gcc -c -Wall helloubuntu.c -o kubuntu.o
```

當構建對象庫或者生成一系列對象文件以備稍後鏈接用時，一條命令即可從多個源碼文件生成對應的對象文件。下面的命令將生成對象文件ubuntu.o, kubuntu.o 與 xubuntu.o：

```
$ gcc -c -Wall ubuntu.c kubuntu.c xubuntu.c
```

多個源文件生成可執行程序

即使多個源碼文件被編譯，GCC編譯器也會自動進行鏈接操作。例如：下面的代碼保存在名為 hellomain.c 的文件中並調用一個名為 sayhello()的函數：

```
/* hellomain.c */
void sayhello(void);
int main(int argc,char *argv[])
{
        sayhello();
        return 0;
}
```

以下代碼保存在名為 sayhello.c 的文件中並定義了 sayhello() 函數：

```
/* sayhello.c */
#include <stdio.h>
void sayhello()
{
        printf("hello, ubuntu\n");/*这里有个小错误，是中文输入法造成的引号使gcc报错*/
}
```

下面的命令將兩個文件分別編譯為對象文件且將其鏈接為可執行程序 hello，並刪除對象文件：

```
$ gcc -Wall hellomain.c sayhello.c -o hello
```

編譯預處理

選項 -E 指示編譯器只進行編譯預處理。下面的命令將預處理源碼文件 helloubuntu.c 並將結果在標準輸出中列出：

```
$ gcc -E helloubuntu.c
```

選項 -o 用來將預處理過的代碼定向到一個文件。像本文一開始給出的後綴列表所給出的，不需經過預處理的C源碼文件保存為後綴為 .i的文件中，這種文件可以這樣來獲得：

```
$ gcc -E helloubuntu.c -o helloubuntu.i
```

生成彙編代碼

選項 -S 指示編譯器生成彙編語言代碼然後結束。下面的命令將由 C 源碼文件 helloubuntu.c 生成彙編語言文件 helloubuntu.s：

```
$ gcc -S helloubuntu.c
```

彙編語言的形式依賴於編譯器的目標平台。如果多個源碼文件被編譯，每個文件將分別產生對應的彙編代碼模塊。

創建靜態庫

靜態庫是編譯器生成的普通的 .o 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。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 。

要構建一個庫，首先要編譯出庫中需要的對象模塊。例如，下面的兩個源碼文件為 hellofirst.c 和 hellosecond.c：

```
/* hellofirst.c */
#include <stdio.h>
void hellofirst()
{
        printf( “The first hello\n” );
}
```

```
/* hellosecond.c */
#include <stdio.h>
void hellosecond()
{
        printf( “The second hello\n” );
}
```

這兩個源碼文件可以用以下命令編譯成對象文件：

```
$ gcc -c -Wall hellofirst.c hellosecond.c
```

程序 ar 配合參數 -r 可以創建一個新庫並將對象文件插入。如果庫不存在的話，參數 -r 將創建一個新的，並將對象模塊添加（如有必要，通過替換）到歸檔文件中。下面的命令將創建一個包含本例中兩個對象模塊的名為 libhello.a 的靜態庫：

```
$ ar -r libhello.a hellofirst.o hellosecond.o
```

現在庫已經構建完成可以使用了。下面的程序 twohellos.c 將調用該庫中的這兩個函數：

```
/* twohellos.c */
void hellofirst(void);
void hellosecond(void);
int main(int argc,char *argv[])
{
        hellofirst();
        hellosecond();
        return 0;
}
```

程序 twohellos 可以通過在命令行中指定庫用一條命令來編譯和鏈接，命令如下：

```
$ gcc -Wall twohellos.c libhello.a -o twohellos
```

靜態庫的命名慣例是名字以三個字母 lib 開頭並以後綴 .a 結束。所有的系統庫都採用這種命名慣例，並且它允許通過 -l(ell) 選項來簡寫命令行中的庫名。下面的命令與先前命令的區別僅在於 gcc 期望的找尋該庫的位置不同：

```
$ gcc -Wall twohellos.c -lhello -o twohellos
```

指定完整的路徑名可使編譯器在給定的目錄中尋找庫。庫名可以指定為絕對路徑（比如 /usr/worklibs/libhello.a）或者相對與當前目錄的路徑（比如 ./lib/libhello.a）。選項 -l 不能具有指定路徑的能力，但是它要求編譯器在系統庫目錄下找尋該庫。

創建共享庫

共享庫是編譯器以一種特殊的方式生成的對象文件的集合。對象文件模塊中所有地址（變量引用或函數調用）都是相對而不是絕對的，這使得共享模塊可以在程序的運行過程中被動態地調用和執行。

要構建一個共享庫，首先要編譯出庫中需要的對象模塊。例如：下面是文件名為 shellofirst.c 和 shellosecond.c 的兩個源碼文件：

```
/* shellofirst.c */
#include <stdio.h>
void shellofirst()
{
        printf( “The first hello from a shared library\n” );
}
/* shellosecond.c */
#include <stdio.h>
void shellosecond()
{
        printf( “The second hello from a shared library\n” );
}
```

要將以上兩個源碼文件編譯成對象文件，可以用下面的命令：

```
$ gcc -c -Wall -fpic shellofirst.c shellosecond.c
```

選項 -c 告訴編譯器只生成 .o 的對象文件。選項 -fpic 使生成的對象模塊採用浮動的（可重定位的）地址。縮微詞 pic 代表“位置無關代碼”（position independent code）。

下面的 gcc 命令將對象文件構建成一個名為 hello.so 的共享庫：

```
$ gcc -Wall -shared shellofirst.o shellosecond.o -o hello.so
```

選項 -o 用來為輸出文件命名，而文件後綴名 .so 告訴編譯器將對象文件鏈接成一個共享庫。通常情況下，鏈接器定位並使用 main() 函數作為程序的入口，但是本例中輸出模塊中沒有這種入口點，為抑制錯誤選項 -shared 是必須的。

編譯器能將後綴為 .c 的文件識別為 C 語言源代碼文件，並知道如何將其編譯成為對象文件。基於這一點，先前的兩條命令我們可以合併為一條;下面的命令直接將模塊編譯並存儲為共享庫：

```
$ gcc -Wall -fpic -shared shellofirst.c shellosecond.c -o hello.so
```

下面的程序，存儲在文件 stwohellos.c 內，是調用共享庫中兩個函數的主程序：

```
/* stwohellos.c */
void shellofirst(void);
void shellosecond(void);
int main(int argc,char *argv[])
{
        shellofirst();
        shellosecond();
        return 0;
}
```

該程序可以用下面的命令編譯並鏈接共享庫：

```
$ gcc -Wall stwohellos.c hello.so -o stwohellos
```

程序 stwohello 已經完成，但要運行它必須讓其能定位到共享庫 hello.so，因為庫中的函數要在程序運行時被加載。需要注意的是，當前工作目錄可能不在共享庫的查找路徑中，因此需要使用如下的命令行設定環境變量LD_LIBRARY_PATH：

```
 $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
```

超越命名慣例

如果環境要求你使用 .c 以外的後綴名來命名你的 C 源碼文件，你可以通過 -x 選項來指定其對應的語言以忽略我們的命名規範。例如，下面的命令將從文件 helloworrld.jxj 編譯 C 語言源代碼並生成可執行文件 helloubuntu：

```
$ gcc -xc helloubuntu.jxj -o helloubuntu
```

通常，在沒有 -x 選項的情況下，任何具有未知後綴名的源碼文件名都被認為是連接器可以識別的選項，並在不做任何更改的情況下傳遞給鏈接器。選項 -x 對其後的所有未知後綴的文件都起作用。例如，下面的命令使 gcc 將 align.zzz 和 types.xxx 都作為 C 源碼文件來處理：

```
$ gcc -c -xc align.zzz types.xxx
```

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

文的匿名用戶和其他的工作基礎上。

- 關於Ubuntu中文
- 免責聲明