程式設計

函式庫
Library

蘇維宗(Wei-Tsung Su)
suwt@au.edu.tw
564D

目標

函式庫介紹

自製靜態函式庫

自製動態函式庫

函式庫簡介

函式庫

函式庫提供一組**具有相關功能的程式集合**給程式設計師使用，其優點是讓程式設計師不需要花時間開發所有功能。(**踩在巨人的肩膀上**)

例如，stdio函式庫提供基本輸入/輸出功能、OpenCV提供影像處理功能、json-c提供JSON格式處理功能、CUnit提供單元測試功能、raylib提供簡單的遊戲製作等。

註:想像自己寫一個`printf()`與`scanf()`函式要花多少時間?

為何要使用函式庫?

從函式庫使用者(應用程式開發者)的角度來看

- 善用函式庫可以讓應用程式的開發過程事半功倍(**專注於自己的創意**)

從函式庫開發者的角度來看

- 販售函式庫整合、維護與支援服務(**不一定要提供原始碼**)
- 希望網路社群高手一起維護函式庫品質(提供原始碼/自由軟體)
- 可模組化程式碼並重複利用(大型專案開發)
- 佛心來的

函式庫的連結方式

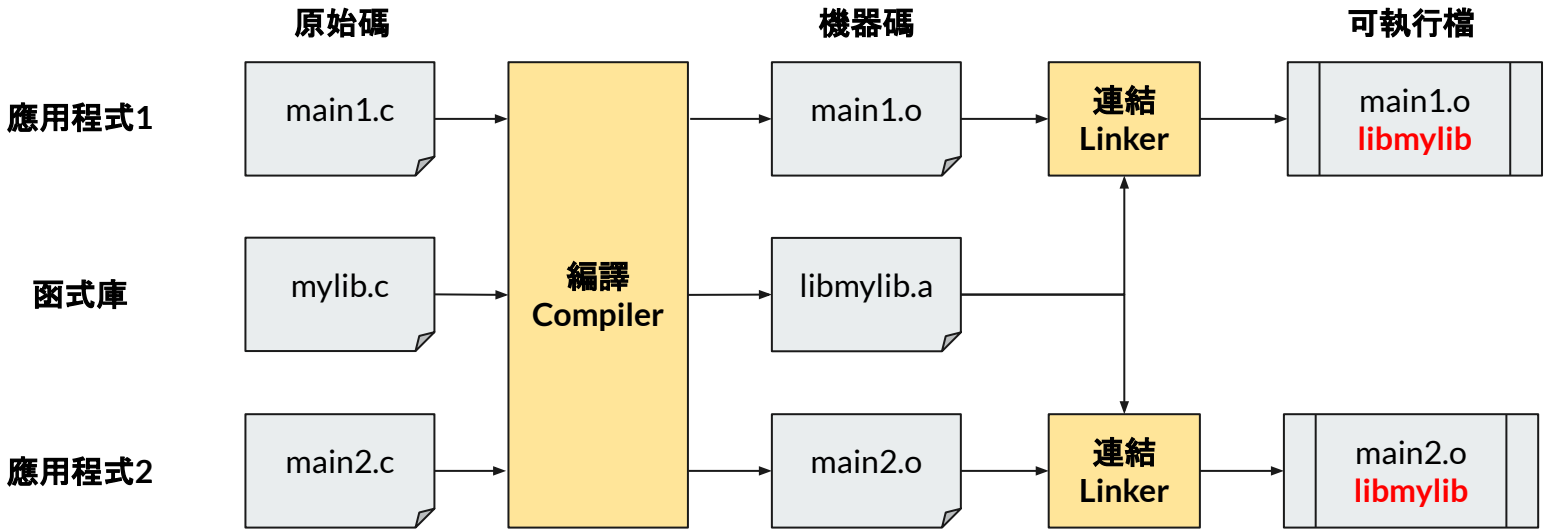
靜態函式庫(static library)

- 在編譯階段(compiler time)將函式庫中需要的部分連結到應用程式

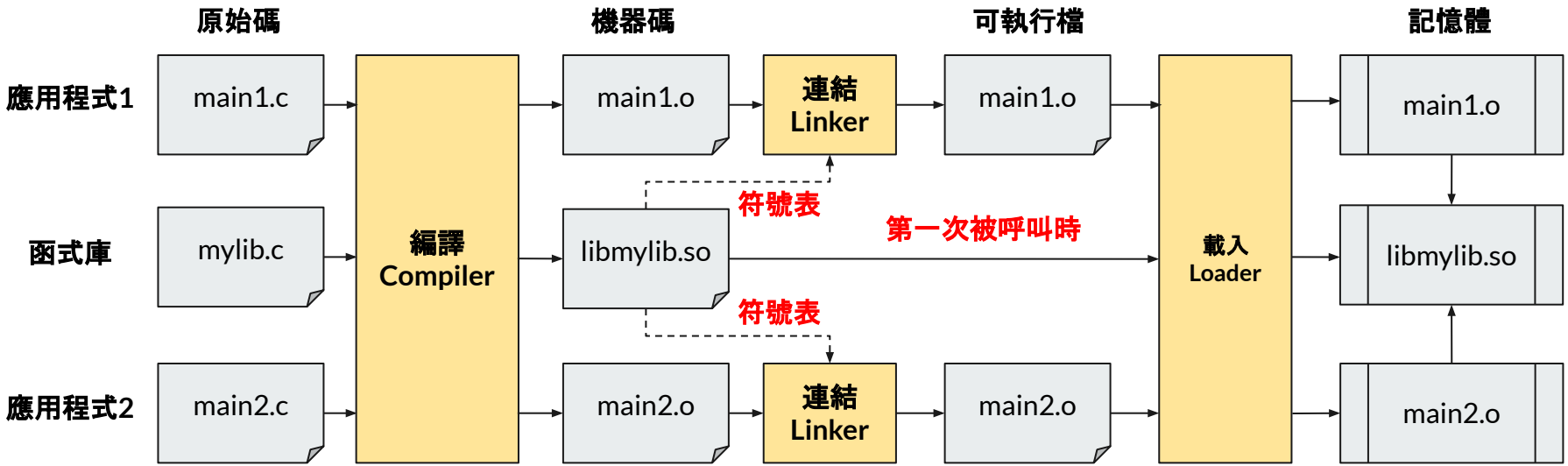
動態函式庫(dynamic library) / 共享函式庫(shared library)

- 在執行階段(run time)將函式庫中需要的部分連結到應用程式
- 由於在執行階段才進行連結，因此應用程式可以共享函式庫

靜態函式庫


原始碼
機器碼
可執行檔
應用程式1
函式庫
應用程式2
main1.c
mylib.c
main2.c
編譯
Compiler
main1.o
libmylib.a
main2.o
連結
Linker
連結
Linker
main1.o
libmylib
main2.o
libmylib

動態函式庫


原始碼
機器碼
可執行檔
記憶體
應用程式1
函式庫
應用程式2
main1.c
mylib.c
main2.c
編譯
Compiler
main1.o
libmylib.so
main2.o
連結
Linker
連結
Linker
符號表
符號表
第一次被呼叫時
main1.o
載入
Loader
main1.o
libmylib.so
main2.o

靜態函式庫 vs. 動態函式庫

靜態函式庫

1. 編譯時期連結
2. 執行檔比較大
3. 執行速度快
4. 函式庫更新時，所有使用此函式庫的應用程式都要重新編譯

動態函式庫

1. 執行時期連結
2. 執行檔比較小(函式庫可共享)
3. 執行速度稍慢(需查詢函式庫位址)
4. 函式庫更新時，只需要更換系統中的動態函式庫

複習：標頭檔(.h檔案)

C語言為何要引用標頭檔(#include<stdio.h>)?

讓編譯器知道程式使用了哪些函式庫

標頭檔內要寫什麼?

全域變數、結構、函式宣告等

下面兩種引用標頭檔的方式有何不同?

```
#include<stdio.h> //標頭檔放在系統預設路徑
#include"mylib.h" //反之，因用時需直接寫出標頭檔路徑
```

自製靜態函式庫

函式庫原始檔

mymath1.c

```
1.   int myadd(int x, int y) {
2.     return x + y;
3.   }
4.   int mysub(int x, int y) {
5.     return x - y;
6.   }
```

mymath.h

```
1.   #ifndef MYMATH_H
2.   #define MYMATH_H
3.   /* 輸入: 兩個整數
4.      回傳: 兩個整數相加的結果 */
5.   int myadd(int, int);
6.   ...
7.   int mysub(int, int);
8.   #endif
```

產生與使用靜態函式庫

產生靜態函式庫

```
$ gcc -c mymath1.c
$ ar rcs libmymath.a mymath1.o
```

編譯程式時未連結靜態函式庫 (會怎樣?)

```
$ gcc main.c
```

編譯程式時連結靜態函式庫

```
$ gcc main.c ./libmymath.a
```

main.c

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.  #include"mymath.h"
4.
5.  int main() {
6.    printf("%d\n", myadd(10, 5));
7.    //printf("%d\n", mymul(10, 5));
8.    return EXIT_SUCCESS;
9.  }
```

函式庫原始檔(增加功能)

mymath2.c

```
1.   int mymul(int x, int y) {
2.     return x * y;
3.   }
4.   float mydiv(int x, int y) {
5.     return (float)x / y;
6.   }
```

mymath.h

```
1.   ...
2.   int myadd(int, int);
3.   ...
4.   int mysub(int, int);
5.   ...
6.   int mymul(int, int);
7.   ...
8.   float mydiv(int, int);
```

產生與使用靜態函式庫

產生靜態函式庫

```
$ gcc -c mymath1.c mymath2.c
$ ar rcs libmymath.a mymath1.o mymath2.o
```

編譯程式時連結靜態函式庫

```
$ gcc main.c ./libmymath.a
```

自製動態函式庫

產生與使用動態函式庫

產生靜態函式庫

```
$ gcc -c -fPIC mymath1.c mymath2.c
$ gcc -shared -o libmymath.so mymath1.o mymath2.c
```

註：Positiion Independent Code(PIC)是讓此程式能放在記憶體內不同位置且能被其他程式呼叫

編譯程式時連結動態函式庫

```
$ gcc main.c -L. -lmymath
```

註：動態函式庫必須被放在系統預設路徑才能被使用(但需要系統管理者)。
註：使用-L選項能指定搜尋動態函式庫的路徑，而-l選項能指定引用的動態函式庫。

練習:學生成績(函式庫)

1. 將學生結構(姓名、程式設計成績、資工導論成績)與相關存取函式撰寫成函式庫
2. 撰寫應用程式引用此函式庫

學生結構如下

```
typedef struct {
  char name[6];
  unsigned char prog, csie;
} student;
```

實作的存取函式如下

```
char* getName(student *s);
unsigned char getProg(student* s);
unsigned char getCsie(student* s);
void setName(student *s, char* n);
void setProg(student* s, unsigned char p);
void setCsie(student* s, unsigned char c);
void printStudent(student* s);
```

Q & A

Computer History Museum, Mt. View, CA