

Hướng dẫn Lab 2.3 – Liên kết tĩnh và liên kết động

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Truyền đối số	Ch2.5 Compile and Linking	Ubuntu 16 / 18

Yêu cầu sinh viên: Hiểu lý thuyết về quá trình biên dịch từ tập tin mã nguồn ra các tập tin đối tượng, biên dịch các thư viện, và bước liên kết chúng với nhau thành một chương trình khả thực thi.

Đánh giá sinh viên: Trả lời các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các Yêu cầu và Bài tập ở cuối hướng dẫn.

Preferences

[1] Chua Hock-Chuan, GCC and Make - Compiling, Linking and Building C/C++ Applications, Đại học Kỹ thuật Nanyang, Singapore, tháng 3-2018

[2] GCC Manual "Using the GNU Compiler Collection (GCC)" @ <http://gcc.gnu.org/onlinedocs>.

Thư viện là một tập hợp các tập tin đối tượng được biên dịch sẵn và có thể được liên kết vào các chương trình của bạn thông qua trình liên kết. Ví dụ là các hàm hệ thống như printf() và sqrt(). Có hai loại thư viện bên ngoài: thư viện tĩnh và thư viện dùng chung.

1. Một thư viện tĩnh có phần mở rộng tệp ".a" (tệp lưu trữ) trong Unix hoặc ".lib" (thư viện) trong Windows. Khi chương trình của bạn được liên kết với thư viện tĩnh, mã máy của các hàm số bên ngoài được sử dụng trong chương trình của bạn sẽ được sao chép vào tập tin thực thi. Một thư viện tĩnh có thể được tạo thông qua chương trình "ar.exe".

2. Thư viện dùng chung có phần mở rộng tệp là ".so" (shared objects) trong Unix hoặc "dll" (dynamic link library) trong Windows. Khi chương trình của bạn được liên kết với thư viện dùng chung, chỉ một bản nhỏ được tạo trong tệp tin thực thi. Trước khi quá trình thực thi bắt đầu, hệ điều hành sẽ tải mã máy cần thiết cho các chức năng bên ngoài - một tiến trình được gọi là liên kết động. Liên kết động làm cho các tệp thực thi nhỏ hơn và tiết kiệm dung lượng ổ đĩa, vì một bản sao của thư viện có thể được chia sẻ giữa nhiều chương trình. Hơn nữa, hầu hết các hệ điều hành cho phép một bản sao của thư viện dùng chung trong bộ nhớ được sử dụng bởi tất cả các chương trình đang chạy, do đó, tiết kiệm bộ nhớ. Mã thư viện dùng chung có thể được nâng cấp mà không cần biên dịch lại chương trình của bạn.

Do lợi thế của liên kết động, theo mặc định, GCC sẽ liên kết đến thư viện dùng chung nếu có. Bạn có thể liệt kê nội dung của thư viện thông qua lệnh "nm filename".

Yêu cầu 1: Tạo một thư viện có 2 hàm sum tính tổng các số nguyên từ 1 đến n, và hàm fac tính giá trị n!. Mỗi hàm viết trên một tệp tin nguồn riêng biệt.

Hướng dẫn: Tải về 2 tệp tin *LAB_IntroOS/LAB_2/task3_1sum.c* và */task3_1fac.c* sau đó biên dịch như sau:

```
> gcc -c task3_1sum.c
> gcc -c task3_1fac.c
```

Tiếp tục thực hiện thư viện libh.a

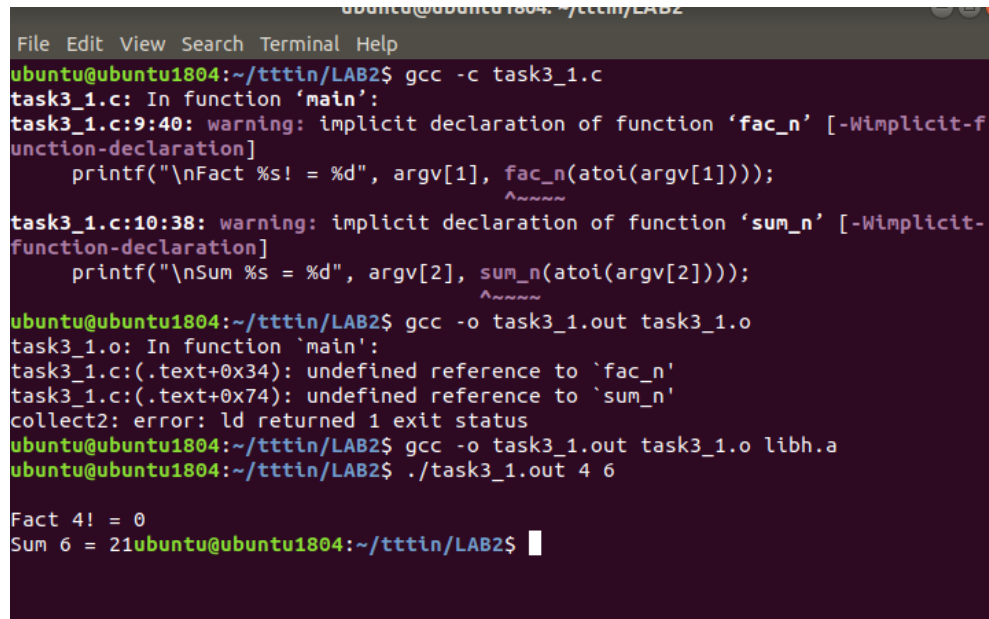
```
> ar cr libh.a task3_1sum.o task3_2fac.o
```

Tải về tệp tin *LAB_IntroOS/LAB_2/task3_1.c* trong đó có gọi 2 hàm đã hiện thực tại các tệp tin .c phía trên (mà bây giờ đã dịch thành thư viện libh.a). Tiếp tục biên dịch

```
> gcc -c task3_1.c
> gcc -o task3_1.out task3_1.o // ERROR
> gcc -o task3_1.out task3_1.o libh.a
```

Chúng ta có thể thực thi tệp tin cuối cùng bằng lệnh, lưu ý rằng tại thời điểm này nếu tệp tin libh.a không còn tồn tại thì cũng không ảnh hưởng đến task3_1.out vì thư viện đã được sao chép vào trong chương trình này.

```
> ./task3_1.out 4 6
```



```
ubuntu@ubuntu1804:~/tttin/LAB2$ gcc -c task3_1.c
task3_1.c: In function 'main':
task3_1.c:9:40: warning: implicit declaration of function 'fac_n' [-Wimplicit-f
unction-declaration]
    printf("\nFact %s! = %d", argv[1], fac_n(atoi(argv[1])));
                                   ^~~~~~
task3_1.c:10:38: warning: implicit declaration of function 'sum_n' [-Wimplicit-
function-declaration]
    printf("\nSum %s = %d", argv[2], sum_n(atoi(argv[2])));
                                   ^~~~~~
ubuntu@ubuntu1804:~/tttin/LAB2$ gcc -o task3_1.out task3_1.o
task3_1.o: In function 'main':
task3_1.c:(.text+0x34): undefined reference to `fac_n'
task3_1.c:(.text+0x74): undefined reference to `sum_n'
collect2: error: ld returned 1 exit status
ubuntu@ubuntu1804:~/tttin/LAB2$ gcc -o task3_1.out task3_1.o libh.a
ubuntu@ubuntu1804:~/tttin/LAB2$ ./task3_1.out 4 6

Fact 4! = 0
Sum 6 = 21ubuntu@ubuntu1804:~/tttin/LAB2$
```

Hình 1. Các lệnh biên dịch liên kết tĩnh libh.a vào task3_1.out

Yêu cầu 2: Tạo một thư viện liên kết động có 2 hàm sum tính tổng các số nguyên từ 1 đến n, và hàm fac tính giá trị n!. Mỗi hàm viết trên một tập tin nguồn riêng biệt.

Hướng dẫn: Dùng lại 2 tập tin *LAB_IntroOS/LAB_2/task3_1sum.c* và */task3_1fac.c* sau đó biên dịch như sau:

```
> gcc -c -fPIC task3_1sum.c
> gcc -c -fPIC task3_1fac.c
```

Tiếp tục thực hiện thư viện libh.a

```
> gcc -shared -fPIC -o libd.a task3_1sum.o task3_1fac.o
```

Tải về tập tin *LAB_IntroOS/LAB_2/task3_2.c*, trong đó có gọi 2 hàm đã hiện thực tại các tập tin .c phía trên (mà bây giờ đã dịch thành thư viện libd.a). Tiếp tục biên dịch

```
> gcc -c task3_2.c
> gcc -o task3_2.out task3_2.o libd.a
```

Và thực thi **task3_2.out**, lúc này hệ thống sẽ báo lỗi “./task3_2.out: error while loading shared libraries: libd.a: cannot open shared object file: No such file or directory”, lý do là thư viện chỉ liên kết chứ không nạp hoàn toàn vào chương trình này. Đây là điểm khác biệt giữa liên kết tĩnh và liên kết động.

Chúng ta cần đặt thư viện libd.a vào thư mục lib

```
> sudo cp libd.a /lib
```

Rồi thực thi lại, chương trình sẽ chạy được.

```
> ./task3_2.out 3 4
```

Bài tập

1. Tiếp tục biên tập tập tin task3_1div.c trong đó chứa hàm void div_n(int n) có chức năng in ra màn hình các ước số của n.
 - a. Biên dịch thư viện libh1.a từ 3 tập tin nguồn .c chứa 3 hàm sum_n, fac_n và div_n.
 - b. Viết tập tin Ex3_1.c liên kết tĩnh với thư viện vừa tạo và gọi 3 hàm trên 3 đối số truyền vào.

```
> ./Ex3_1.out 2 3 4
> Sum 2 = 2
> Fac 3! = 6
> Divisor 4 = 1, 2, 4
```

2. Với bài tập 1, hãy xây dựng thư viện libd1.a liên kết động, sao chép nó vào thư mục /lib rồi biên dịch tập tin Ex3_2.c liên kết động với thư viện vừa tạo và thực thi.