FACULTY OF INFORMATION TECHNOLOGY

>>> COURSE MATERIAL <<<

INTRODUCTION TO OPERATING SYSTEM

Course ID 502047

Lab Part 2 - Compiling with GCC

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Hello Word	Ch1: Giao diện dòng lệnh	Sử dụng image Ubuntu 14
Biên dịch bằng gcc		
Truyền đối số	Ch2: Compile and Linking	
Liên kết động và liên kết tĩnh	Ch2: Compile and Linking	
~~Tạo tiến trình~~	~~Ch3: Process Creation~~	
Makefile		SV tự học

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

Preferences

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

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

[3] GNU 'make' manual @ http://www.gnu.org/software/make/manual/make.html.

[4] Robert Mecklenburg, "Managing Projects with GNU Make", 3rd Edition, 2004.

1. Cài đặt và kiểm tra GCC

GNU Toolchain, bao gồm GCC, luôn đính kèm trong Ubuntu như là trình biên dịch tiêu chuẩn.

Để kiểm tra phiên bản GCC, gõ:

```
$ gcc --version
gcc (GCC) 6.4.0
```

Một số thông tin mở rộng có thể tìm thấy bằng cách sử dụng tuỳ chọn -v

```
$ gcc -v
```

Hoặc tìm kiếm giúp đỡ bằng tuỳ chọn -help

```
$ gcc --help
```

Để đọc hướng dẫn của GCC, gõ:

```
$ man gcc
// Press space key for next page, or 'q' to quit.
```

Để xuất hướng dẫn ra một tập tin văn bản, gõ:

```
$ man gcc | col -b > gcc.txt
```

Ngoài ra, hướng dẫn có thể tham khảo ở http://linux.die.net/man/1/gcc.

Hoặc tìm kiếm ở thư mục "usr/share/man/man1".

```
$ whereis gcc
gcc: /usr/bin/gcc.exe /usr/lib/gcc /usr/share/man/man1/gcc.1.gz
```

2. Biên dịch chương trình đầu tiên

Trình biên dịch GNU C là gcc và cho C++ là g++.

Tạo ra một tập tin mã nguồn hello.c và nhập nội dung sau đây.

```
1       // hello.c
2       #include <stdio.h>
3
4       int main() {
5           printf("Hello, world!\n");
6           return 0;
7       }
```

Ví dụ 1. Hello World

Để biên dịch tập tin hello.c vừa tạo ra, sử dụng lệnh:

```
> gcc hello.c
  // Compile and link source file hello.c into executable a
(Unixes)
```

Tên chương trình đầu ra mặc định là "a.out" (Unixes and Mac OS X).

Để chạy chương trình vừa tạo ra, gõ:

```
// (Unixes) In Bash Shell - include the current path (./)
$ chmod a+x a.out
$ ./a.out
```

Ghi chú cho Unix và Bash Shell:

- Trong Bash shell, PATH mặc định không bao gồm thư mục làm việc hiện tại. Do đó, bạn cần bao gồm đường dẫn hiện tại (./) trong lệnh.
- Cần gõ đầy đủ tên chương trình bao gồm phần mở rộng tập tin, nếu có, tức là, "./a.out".
- Trong Unix, tập tin đầu ra có thể là "a.out" hoặc đơn giản là "a". Hơn nữa, bạn cần gán chế độ tập tin khả thực thi (x) cho "a.out", thông qua lệnh "chmod a + x tên tập tin" (thêm chế độ tập tin khả thực thi "+ x" cho tất cả người dùng "a + x").

Để chỉ định tên tệp đầu ra, sử dụng tùy chọn -o:

```
$ gcc -o hello hello.c
$ chmod a+x hello
$ ./hello
```

Ghi chú cho Unix

- Trong Unix, chúng ta thường bỏ qua phần mở rộng tập tin .exe (chỉ dành cho Windows) và chỉ cần đặt tên tập tin thực thi là hello.out, hoặc không cần phần mở rộng, là hello cũng được.
- Cần gán chế độ tập tin thực thi thông qua lệnh "chmod a + x hello".
- Chúng ta sử dụng g++ để biên dịch chương trình C++ (hiếm gặp trong khoá học này).

Một số tuỳ chọn với GCC

Các tuỳ chọn thường dùng có thể kể đến là:

```
$ g++ -Wall -g -o Hello.exe Hello.cpp
```

- -o: chỉ định tên tập tin chương trình đầu ra.
- -Wall: in ra tất cả thông báo Warning.
- -g: tạo thêm thông tin gỡ lỗi tượng trưng để sử dụng với trình gỡ lỗi gdb.

Biên dịch và liên kết tách biệt

Lệnh nêu trên biên dịch tập tin mã nguồn thành một chương trình đầu ra thông qua một bước duy nhất. Chúng ta có thể tách biệt làm 2 giai đoạn: **biên dịch** (các) tập tin mã nguồn thành (các) tập tin đối tượng và **liên kết** (các) tập tin đối tượng cùng với thư viện hệ thống để cho ra tập tin chương trình cuối cùng bằng các tuỳ chọn như sau:

```
// Bước 1 với tuỳ chọn -c
> gcc -c -Wall -g Hello.cpp
// Bước 2 với tuỳ chọn -o
> gcc -g -o Hello.exe Hello.o
```

Các tùy chọn là:

- -c: Biên dịch thành tập tin đối tượng "Hello.o". Theo mặc định, tập tin đối tượng có cùng tên với tập tin mã nguồn và có phần mở rộng là ".o" (không cần chỉ định tùy chọn -o). Nó không liên kết với các tập tin đối tượng khác nếu có và cũng không liên quan gì đến thư viện hệ thống.
- Liên kết được thực hiện với đầu vào là các tập tin đối tượng ".o" (chứ không phải là ".c"). GCC sử dụng một chương trình liên kết riêng để thực hiện liên kết.

Biên dịch và liên kết nhiều tập tin mã nguồn

Giả sử rằng chương trình của bạn có hai tập tin mã nguồn: file1.cpp, file2.cpp. Chúng có thể được biên dịch bằng lệnh:

```
> g++ -o myprog.exe file1.cpp file2.cpp
```

Tuy nhiên, chúng ta nên biên dịch từng tập tin mã nguồn thành tập tin đối tượng và liên kết chúng với nhau trong giai đoạn sau. Trong trường hợp này, các tập tin mã nguồn nào có thay đổi mới cần phải biên dịch lại.

```
> gcc -c file1.cpp
> gcc -c file2.cpp
> gcc -o myprog.exe file1.o file2.o
```

3. Truyền đối số từ lệnh gọi

Tạo ra một tập tin mã nguồn para.c và nhập nội dung sau đây.

```
1     // para.c
2     #include <stdio.h>
3
4     int main(int argc, char ** argv) {
5         printf("Number  of  arguments  %i  n",  argc);
6         int i=0;
7         for(i=0; i<argc; i++)
8             prinf("Argument %s\t", argv[i];
9
10        return 0;
11    }
```

Ví dụ 2. Arguments from function call

```
> gcc -c para.o para.c
> gcc -o main.out para.o
> ./main.out a 12 test
```

Khi chạy lời gọi trên, argc = 4 và véc tơ argu có giá trị là argv = [main.out, a, 12, test]

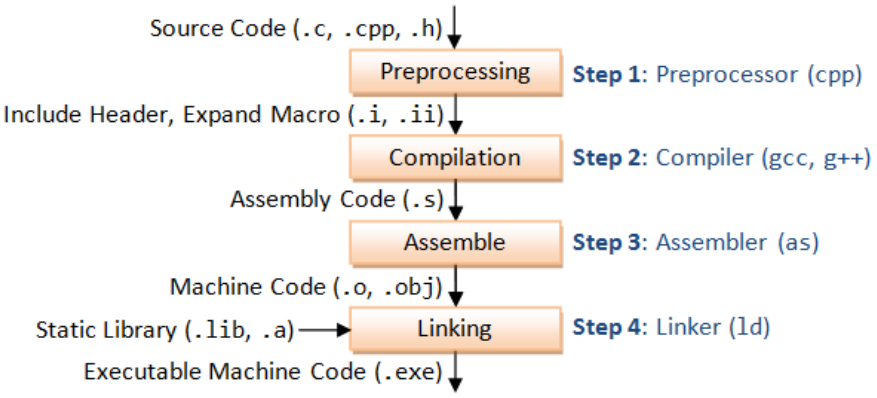
Lưu ý: 12 bên trên là một chuỗi kí tự, để lấy giá trị 12 chúng ta dùng hàm atoi(argv[2]).

4. Biên dịch thành thư viện chia sẻ

Phần này sinh viên dùng để tham khảo chuyên sâu, sinh viên có thể đi đến ngay phần 5 và trở lại đọc phần 4 khi đã hiểu qui trình biên dịch.

Để biên dịch và liên kết chương trình C / C ++ vào một thư viện dùng chung (".so" trong Unix), sử dụng tuỳ chọn -share.

Quá trình biên dịch của GCC



Hình 1. Quá trình biên dịch và liên kết trong GCC

GCC biên dịch chương trình C / C ++ thành khả thực thi qua 4 bước như *Hình 1. Quá trình biên dịch và liên kết trong GCC*. Ví dụ: "gcc -o hello.exe hello.c" được thực hiện như sau:

1. Tiền xử lý: thông qua Bộ tiền xử lý GNU C (cpp.exe), bao gồm các tiêu đề (#include) và mở rộng các macro (#define).

```
> cpp hello.c > hello.i
```

Tập tin trung gian "hello.i" chứa mã nguồn mở rộng.

2. Biên dịch: Trình biên dịch biên dịch mã nguồn đã được tiền xử lý thành mã hợp ngữ cho một bộ vi xử lý cụ thể.

```
> gcc -S hello.i
```

Tùy chọn -S chỉ định để tạo mã hợp ngữ, thay vì tạo mã tập tin đối tượng. Tập tin kết quả trung gian sau bước này là "hello.s".

3. Tập hợp: Trình biên dịch (as) chuyển đổi mã hợp ngữ thành mã máy trong tập tin đối tượng "hello.o".

```
> as -o hello.o hello.s
```

4. Liên kết: Cuối cùng, trình liên kết (ld) liên kết mã đối tượng với mã thư viện để tạo thành tập tin thực thi "hello.exe".

```
> ld -o hello.exe hello.o ...libraries...
```

Xem chi tiết biên dịch

Quy trình biên dịch chi tiết có thể hiển thị bằng cách bật tùy chọn -v (verbose).

```
> gcc -v -o hello.exe hello.c
```

5. Biên dịch liên kết tĩnh và liên kết động

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 bảng nhỏ được tạo trong tập 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 quá 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*”.

Ví dụ liên kết tĩnh: Tạo các tập tin hello1.c và hello2.c như sau:

```
1       // hello1.c
2       #include <stdio.h>
3
4       void hello_1(int i) {
5           printf("Hello, parameter 1 = %d\n", i);
6       }
```

```
1       // hello2.c
2       #include <stdio.h>
3
4       void hello_2(int i) {
5           printf("Hello, parameter 2 = %d\n", i);
6       }
```

Tạo thư viện liên kết tĩnh libh.a từ 2 file hello1.c và hello2.c được tiến hành như sau:

Bước 1: Biên dịch và tạo file object

```
$ gcc -c hello1.c hello2.c
```

Bước 2: Dùng lệnh ar để tạo thư viện tĩnh tên libh.a

```
$ ar cr libh.a hello1.o hello2.o
```

Bước 3: Dùng lệnh nm để xem kết quả

```
$ nm libh.a
```

Dùng lệnh file để xem libh là file gì

```
$ file libh.a
```

Tạo hàm main có sử dụng hàm trong thư viện libh.a như sau:

```
// main.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char ** argv) {
    int i = atoi(argv[1]);
    int k = atoi(argv[2]);
    hello_1(i);  hello_2(k);
}
```

- Nếu chúng ta biên dịch mà không liên kết đến thư viện tĩnh thì sẽ bị lỗi "undefined reference to 'hello1' …"

```
$ gcc -c main.c
$ gcc -o main.o
$ gcc -o main.out main.o
```

- Biên dịch có liên kết đến thư viện tĩnh như sau:

```
$ gcc -c main.c
$ gcc -o main.o
$ gcc -o main.out main.o libh.a
$ ./main.out   //Err: Segmentation fault (core dumped)
$ ./main.out 5 6
Hello, parameter 1 = 5
Hello, parameter 2 = 6
```

Ví dụ liên kết động

Để tạo thư viện liên kết động libd.a từ 2 tập tin hello1.c và hello2.c, ta sử dụng các tuỳ chọn -fPIC và -share như sau:

```
$ gcc -c -fPIC hello1.c hello2.c
$ gcc -shared -fPIC -o libd.a hello1.o hello2.o
```

Để sử dụng thư viện vừa tạo ra, chúng ta phải đưa nó và thư mục /lib bằng lệnh sao chép.

```
$ sudo cp libd.a /lib
$ gcc -c main.c
$ gcc -o main.out2 main.o libd.a
$ ./main.out 2 3
```

Bài tập

1. Viết chương trình sau cho khi truyền đối số n vào thì xuất ra tổng S = 1 + 2 + …. + n
    a. Báo lỗi nếu lời gọi có đối số không phải là một số nguyên dương.
    b. Báo lỗi nếu có nhiều hơn 2 đối số (là main.out và n).

```
> ./main.out 8
> S = 36
> ./main.out abc
> Doi so khong phai la so nguyen duong
> ./main.out 8  19  ab
> Co qua nhieu doi so
```

2. Viết chương trình truyền vào một số nguyên, và in ra dãy các ước số của số nguyên này.
    a. Báo lỗi nếu đối số không phải là số nguyên, hoặc thừa đối số.
    b. Phân tích số nguyên đã truyền vào thành thừa số nguyên tố. (bài tập nâng cao)

```
> ./main.out 12
> Cac uoc so cua 12 la 1, 2, 3, 4, 6, 12
> 12 = 2^2*3
```

```
#include <iostream>
using namespace std;

int main(){
  int n;
  cout << "\nNhap n = ";
  do{
    cin >> n;
    if(n <= 0){
      cout << "\nNhap lai n = ";
    }
  }while(n <= 0);

  for(int i = 1;i <= n;i++){
    if(n % i == 0){
```

```
        cout << i << " ";
      }
    }
}
```

```
#include <stdio.h>

int main(){
   int n;
   printf("\nNhap n = ");
   scanf("%d", &n);
   int dem;

   for(int i = 2; i <= n; i++){
      dem = 0;
      while(n % i == 0){
         ++dem;
         n /= i;
      }
      if(dem){
         if(dem > 1) printf("%d^%d", i, dem);
         else printf("%d", i);
         if(n > i){
            printf(" * ");
         }
      }
   }

}
```

```
        // Chuong trinh truyen vao so nguyen, in ra day cac uoc so cua so nguyen nay

        #include <stdio.h>
        #include <stdlib.h>
        #include <stdio.h>

        int main(int argc, char ** argv) {
            printf("Number of arguments %i n", argc);
            int i=0;
            int check = argc;
            int number = atoi(argv[1]);
            if (check > 2)
                printf("Thua doi so\n");
            else if (number <=0)
                printf("Doi so khong phai la so nguyen duong");
```

```
    else {
        int dem;
            for(int i = 2; i <= n; i++){
            dem = 0;
            while(n % i == 0){
              ++dem;
              n /= i;
            }
            if(dem){
               if(dem > 1)
                   printf("%d^%d", i, dem);
               else
                   printf("%d", i);
               if(n > i){
                   printf(" * ");
               }
            }
        }
    return 0;
}
```

3. Viết chương trình truyền vào một danh sách số nguyên, và in ra dãy số này theo thứ tự tăng dần.
    a. Bỏ qua các đối số không phải là số nguyên.
    b. Hãy áp dụng các thuật toán sắp xếp đã học. (bài tập về nhà).

```
> ./main.out 8 3 1 ab -12  ls
> Day tang là -12 1 3 8
```

4. Viết các tập tin add.c, sub.c lần lượt chứa 2 hàm số `int add(int a, int b)` và `int sub(int a, int b)`.

    a. Xây dựng một thư viện liên kết tĩnh từ 2 tập tin add.c và sub.c
    b. Xây dựng một thư viện liên kết động từ 2 tập tin add.c và sub.c

Lần lượt sử dụng các thư viện ở a. và b., viết hàm main truyền vào hai số nguyên và dấu phép tính "+" hay "-", và in ra kết quả tương ứng. Chương trình báo lỗi nếu các đối số truyền vào không đúng theo qui tắc.

```
> ./main.out 1 2 +
> Ket qua: 3
> ./main.out 3 4 -
> Ket qua: -1
> ./main.out 1 2 *
> Doi so truyen khong dung.
```