

HỆ ĐIỀU HÀNH

TUẦN 3

COMPILER VỚI GCC VÀ G++

HƯỚNG DẪN

1. Chương trình trên Linux

- Để có thể viết chương trình trên Linux, chúng ta cần phải nắm rõ 1 số vị trí tài nguyên để xây dựng chương trình như trình biên dịch, tập tin thư viện, các tập tin tiêu đề (header), các tập tin chương trình sau khi biên dịch, ...
- Ta có thể dùng công cụ **gcc** cho các chương trình viết bằng C chuẩn hoặc **g++** cho các chương trình viết bằng C++
- Trình biên dịch **gcc** thường được đặt trong thư mục **/usr/bin** hoặc **/usr/local/bin** (kiểm tra bằng lệnh **which gcc**). Tuy nhiên, khi biên dịch, **gcc** cần đến rất nhiều tập tin hỗ trợ nằm trong những thư mục khác nhau như những tập tin tiêu đề (header) của C thường nằm trong thư mục **/usr/include** hay **/usr/local/include**. Các tập tin thư viện liên kết thường được **gcc** tìm trong thư mục **/lib** hoặc **/usr/local/lib**. Các thư viện chuẩn của **gcc** thường đặt trong thư mục **/usr/lib/gcc-lib**.

Chương trình sau khi biên dịch ra tập tin thực thi (dạng nhị phân) có thể đặt bất cứ vị trí nào trong hệ thống.

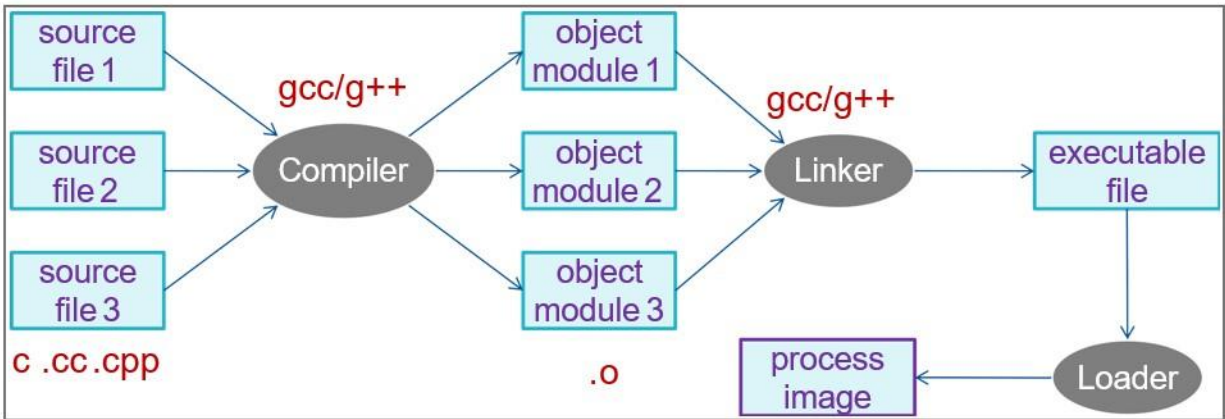
2. Các tập tin tiêu đề (header)

- Các tập tin tiêu đề trong C thường định nghĩa hàm và khai báo cần thiết cho quá trình biên dịch. Hầu hết các chương trình trên Linux khi biên dịch sử dụng các tập tin tiêu đề trong thư mục **/usr/include** hoặc các thư mục con bên trong thư mục này, ví dụ: **/usr/include/sys**. Một số khác được trình biên dịch dò tìm mặc định như **/usr/include/X11** đối với các khai báo hàm lập trình đồ họa X-Window, hoặc **/usr/include/g++-2** đối với trình biên dịch GNU **g++**.

Tuy nhiên, nếu chúng ta có các tập tin tiêu đề của riêng mình trong một thư mục khác thư mục mặc định của hệ thống thì chúng ta có thể chỉ rõ tường minh đường dẫn đến thư mục khi biên dịch bằng tùy chọn **-I**, ví dụ: **\$ gcc -I/usr/mypro/include test.c -o test**

- Khi chúng ta sử dụng một hàm nào đó của thư viện hệ thống trong chương trình C, ngoài việc phải biết khai báo nguyên mẫu của hàm, chúng ta cần phải biết hàm này được định nghĩa trong tập tin tiêu đề nào. Trình **man** sẽ cung cấp cho chúng ta các thông tin này rất chi tiết. Ví dụ, khi dùng **man** để tham khảo thông tin về hàm **kill()**, chúng ta sẽ thấy rằng cần phải khai báo 2 tập tin tiêu đề là **types.h** và **signal.h**.

Quá trình biên dịch của 1 file:



Trước khi chạy chương trình, ta phải thực hiện kiểm tra phiên bản của gcc hoặc g++

```
$ gcc --version
```

Hoặc

```
$ g++ --version
```

Trong trường hợp nếu chưa có gcc hoặc g++ ta thực hiện cài đặt như sau:

```
$ sudo apt-get install gcc
```

Hoặc

```
$ sudo apt-get install g++
```

Tuy nhiên, trong một số tình huống, với các dòng Ubuntu quá cũ, ta phải thực hiện update cho Ubuntu trước bằng câu lệnh:

```
$ sudo apt-get update
```

Sau khi đã chắc chắn cài đặt xong, ta có thể cài thêm bộ build-essential để chắc chắn các thư viện chuẩn đã được đưa vào sử dụng.

```
$ sudo apt-get install build-essential
```

Tới đây, ta đã bắt đầu chạy các chương trình trên Ubuntu.

Ví dụ 1: Giả sử ta soạn thảo một hàm hello đơn giản để in ra màn hình chữ Hello world...!

Được lưu trong file hello.c

```
#include <stdio.h>
void hello()
{
    printf("Hello world...! \n");
}
```

Ta bắt đầu compile chương trình như sau:

Ta thực hiện qua 2 bước:

Bước 1:

```
$ gcc -c hello.c
```

Tại bước này, 1 file hello.o sẽ được sinh ra. Đây là file object sinh ra tương ứng với file hello.c

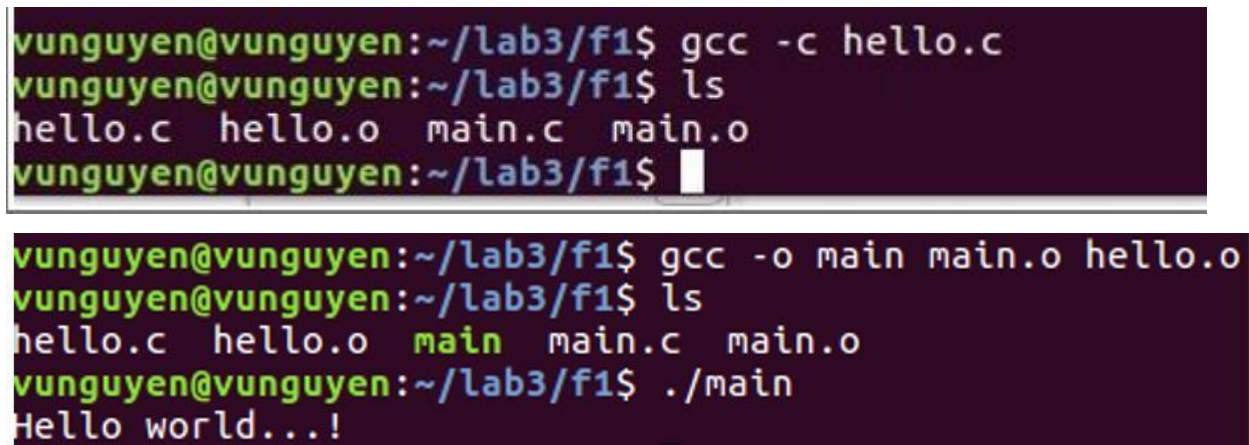
Bước 2:

```
$ gcc -o hello.out hello.o
```

Tại đây, 1 file hello.out sẽ được sinh ra. Đây chính là file thực thi chương trình. Khi cần chạy chương trình, ta sẽ gọi:

```
$ ./hello.out
```

Đoạn chương trình sẽ thực hiện yêu cầu.



```
vunguyen@vunguyen:~/lab3/f1$ gcc -c hello.c
vunguyen@vunguyen:~/lab3/f1$ ls
hello.c hello.o main.c main.o
vunguyen@vunguyen:~/lab3/f1$

vunguyen@vunguyen:~/lab3/f1$ gcc -o main main.o hello.o
vunguyen@vunguyen:~/lab3/f1$ ls
hello.c hello.o main main.c main.o
vunguyen@vunguyen:~/lab3/f1$ ./main
Hello world...!
```

Tương tự, nếu đoạn code chạy bằng C++, ta có thể thực hiện bằng g++.

Truyền đối số từ command.

Ta có thể truyền đối số từ dòng lệnh vào chương trình, tuy nhiên để thực hiện truyền đối số vào chương trình ta phải sử dụng cách thức:

```
int main (int argc, char **argv)
```

Với **argc** là số lượng đối số truyền vào.

Và **argv** là 1 mảng các giá trị đối số truyền vào.

Ví dụ 2:

```
#include <stdio.h>
int main(int argc, char ** argv)
{
    printf("so doi so truyen vao: %i\n", argc);
    printf("gia tri cac doi so: ");
    int i=0;
    for(int i=0; i<argc; i++)
        printf("%s\t", argv[i]);
    printf("\n");
}
```

```
vunguyen@vunguyen:~/lab3/f1$ gcc -c main.c
vunguyen@vunguyen:~/lab3/f1$ gcc -o main.out main.o
vunguyen@vunguyen:~/lab3/f1$ ./main.out 2 test 3 os 5
so doi so truyen vao: 6
gia tri cac doi so: ./main.out 2 test 3 os 5
```

Trong tình huống trên khi ta gọi thực thi main.out ta truyền các đối số là 2 test 3 os 5.

Có tất cả 5 đối số, tuy nhiên chương trình sẽ tính luôn ./main.out là đối số đầu tiên nên giá trị argc sẽ là 6.

Các giá trị lần lượt sẽ 6 đối số như hình.

BÀI THỰC HÀNH

Bài 1

Viết chương trình sau cho khi truyền đối số n vào thì xuất ra tổng $S = 1 + 2 + \dots + n$

- Báo lỗi nếu lời gọi có đối số không phải là một số nguyên dương.
- 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
```

Bài 2

Viết chương trình truyền vào một danh sách số nguyên, và in ra dãy số này theo thứ tự tăng dần.

- Bỏ qua các đối số không phải là số nguyên.
- Hãy áp dụng các thuật toán sắp xếp đã học.

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

HẾT