Thực hành HĐH MNM
Lập trình C/C++ trên Linux

Trịnh Tấn Đạt

Tuần 10

http://sites.google.com/site/ttdat88

Nội dung

- Compiler
- Cài đặt gcc/g++
- IDE thông dụng
- Trình biên dịch GNU C/C++
- Bài tập

Compiler

- Compiler (trình biên dịch) là một chương trình giúp chuyển các đoạn mã mà ta viết bằng một ngôn ngữ nào đó thành một dãy mã ngôn ngữ máy tính (bit).
- Ở Windows trong bộ Visual Studio đã tích hợp sẵn bộ compiler cho ngôn ngữ như C, C++, C# ... Ở Ubuntu cần phải cài riêng biệt .
- Bộ trình dịch GNU là một bộ complier khá nổi tiếng còn được gọi là **GCC** (Viết tắc của GNU Collection Compiler) hoạt động trên nhiều nền tảng từ Windows, Linux, Unix, BDS,... và hỗ trợ các ngôn ngữ : C, C++ (G++), Java (GCJ), Ada (GNAT), Objective-C, Objective-C++, và Fortran.

Cài đặt gcc/g++

- Text editor: **gedit**, nano, … để soạn thảo code
- Compiler: **gcc** dùng cho C hoặc **g++** dùng cho C++
  - Cài đặt : sudo apt install gcc
  - Hoặc có thể cài đặt gcc, g++ và các công cụ hỗ trợ khác (i.e. Make) thông qua package build-essential

sudo apt install build-essential

Cài đặt gcc/g++

- Sau khi cài đặt xong, có thể kiểm tra version của gcc hoặc g++
- Open terminal, type

gcc --version

g++ --version

```
dattt@lythuongkiet:~$ g++ --version
g++ (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

which gcc

which g++

- Trong lúc cài đặt nếu bị lỗi "Could not get lock /var/lib/dpkg/lock-frontend" do vài nguyên nhân :
  - 'Synaptic Package Manager' or 'Software Updater' is open.
  - Some apt command is running in Terminal.
  - Some apt process is running in background
- Fix :

```
sudo killall apt apt-get

sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock*

sudo dpkg --configure -a
sudo apt update
```

- Ví dụ: chương trình C
- Dùng gedit tạo file hello.c (file *.c)

```
#include <stdio.h>
int main()
{
        printf("Hello, World!\n");
        return 0;
}
```

- Compile & run : open terminal, cd <đường dẫn chứa file *.c>

gcc hello.c	gcc hello.c -o hello
./a.out	./hello

-o : flag thể hiện tên của tập tin output. Nếu không đặt tên cho file đầu ra thì theo truyền thống, gcc sẽ tạo file có tên là a.out

- Ví dụ: chương trình C ++
- Dùng gedit tạo file Test.cpp (file ***.cpp** hoặc ***. C**)

```
#include <iostream>
#include <string>
using namespace std;
int main() {
        string s;
        cout<< “Input chuoi" << endl;
        getline(cin,s);
        cout << s << endl;
        cout << s.size() << endl;
        return 0;
}
```

- Compile & run: open terminal

```
g++ Test.c -o Test
./Test
```

- Ví dụ: chương trình C ++ (tao thêm file header)
- Dùng gedit tạo file InputArray.h , InputArray.cpp, Main.cpp

InputArray.h

```
#ifndef __INPUTARRAY_H
#define __INPUTARRAY_H
#include <vector>
#include <iostream>
using namespace std;
void Input(vector<int> &);
void Output(vector<int> );
#endif
```

hoặc InputArray.h

```
#pragma once
#include <vector>
#include <iostream>
using namespace std;
void Input(vector<int> &);
void Output(vector<int> );
```

InputArray.cpp

```
#include "InputArray.h"
void Input(vector<int> &Arr)
 {  Arr.push_back(10);
    Arr.push_back(1);
    Arr.push_back(100); }
 void Output(vector<int> Arr)
{
    cout << "The vector elements are : ";
    for(int i=0; i < Arr.size(); i++)
          cout << Arr[i] << ' '; cout << endl;
}
```

- Ví dụ: chương trình C ++ (tao thêm file header)
- Dùng gedit tạo file InputArray.h , InputArray.cpp, Main.cpp

Main.cpp

```
#include "InputArray.h"
int main()
{
      vector <int> A;
      Input(A);
      Output(A);
       return 0;
}
```

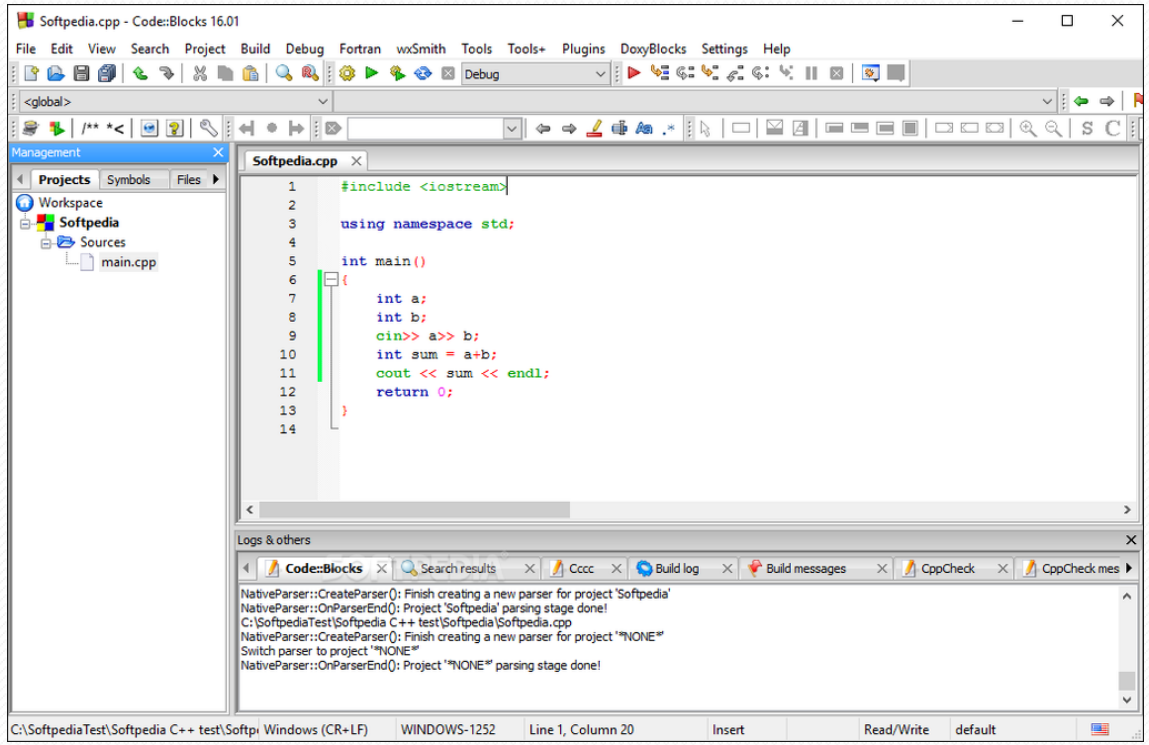
Compile & run : open terminal, cd <đường dẫn chứa file *.h và *.cpp>

g++ InputArray.cpp Main.cpp -o Main

./Main

IDE

- Integrated Development Environment (IDE) là một môi trường tích hợp bao gồm nhiều công cụ khác nhau hỗ trợ việc lập trình C/C++ trên nền tảng Linux,
  - code editor; debugger; simulator; …
- Các IDE thông dụng: Visual Studio Code, Eclipse, Code::Blocks, Qt Createrm, Geany,…



Ví dụ: giao diện Code::Blocks

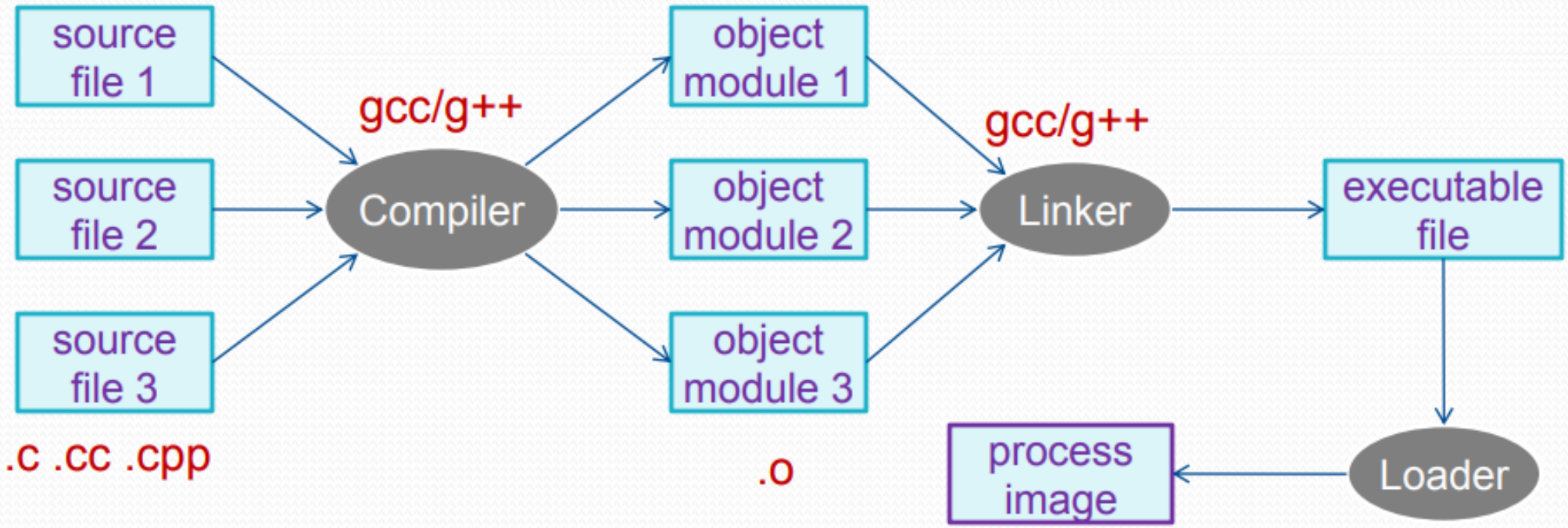
Cài đặt thông qua **terminal** hoặc **Ubuntu Software Center**

Dùng terminal:

sudo apt-get update

sudo apt-get install codeblocks g++

Process


source file 1
source file 2
source file 3
.c .cc .cpp
gcc/g++
Compiler
object module 1
object module 2
object module 3
.o
gcc/g++
Linker
executable file
Loader
process image

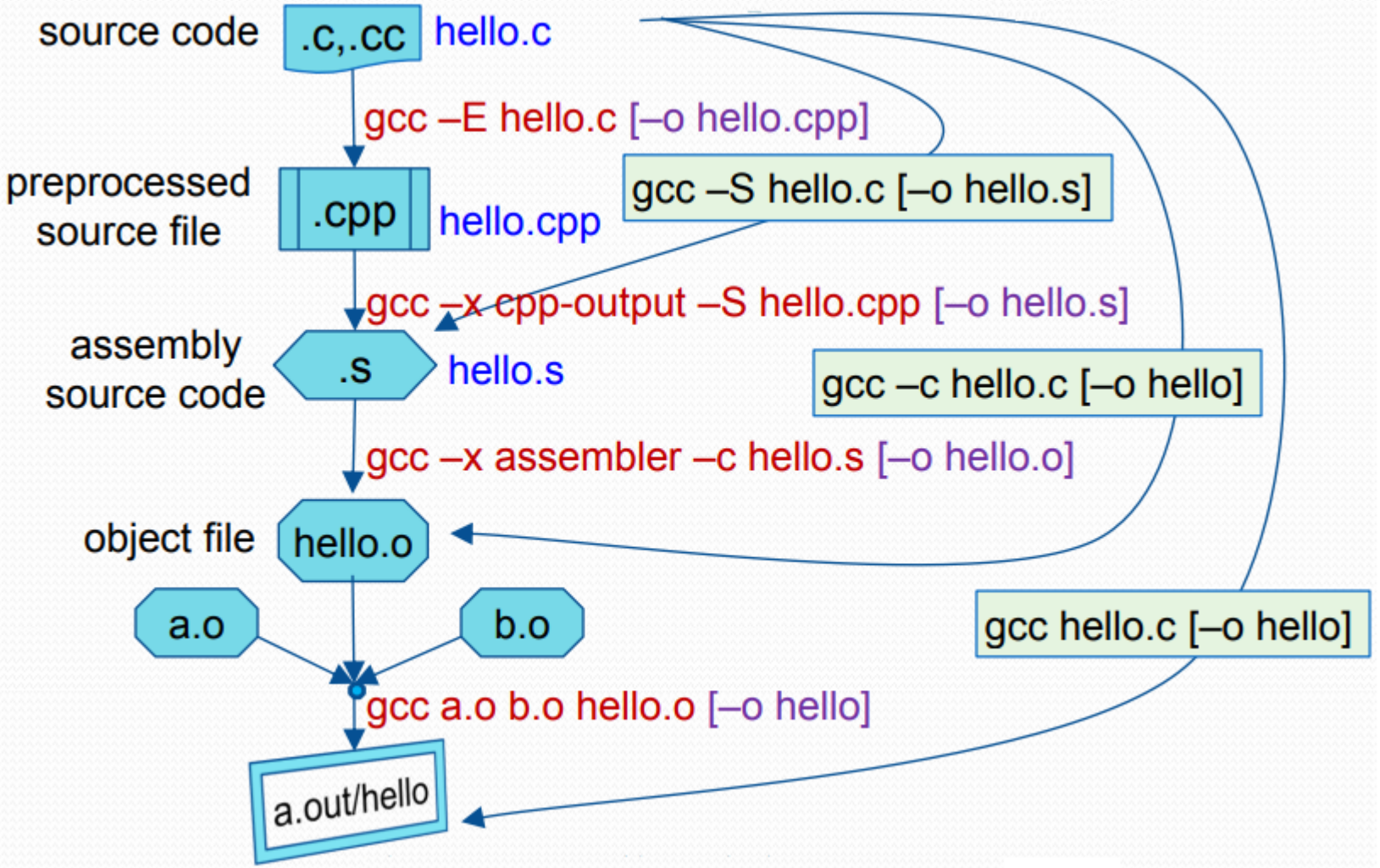
Trình biên dịch GNU C/C++

- Quá trình biên dịch thành file thực thi gồm 4 giai đoạn theo thứ tự như sau:

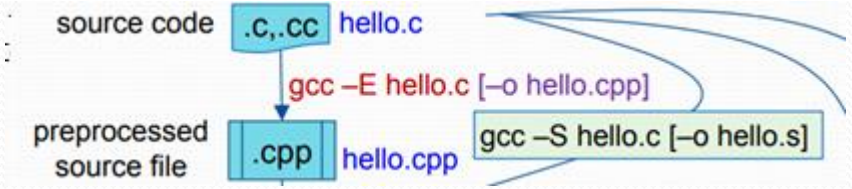
1. preprocessing (tiền xử lý)
2. compilation (biên dịch)
3. assembly (hợp dịch)
4. linking (liên kết)

- Ba bước 1, 2, 3 chủ yếu làm việc với một file đầu vào
- Bước 4 có thể liên kết nhiều object module liên quan để tạo thành file thực thi nhị phân (executable binary)
- Lập trình viên có thể can thiệp vào từng bước ở trên

Trình biên dịch GNU C/C++



Trình biên dịch GNU C/C++



- Pha 1: Preprocessor, giai đoạn tiền xử lý (tạo ra preprocessed file). Tất cả chỉ thị tiền xử lý (Preprocessor directive) bắt đầu với #
  - Thay thế tất cả các macro được định nghĩa trong chương trình
  - Thay thế các câu lệnh include.
  - Để trình biên dịch dừng lại ở pha này ta thêm chỉ thị dịch **–E**.

Ví dụ: tạo file test.c

```
#include <stdio.h>
#define PI 3.14159
int main() {
    printf("Hello World\n");
    printf("Gia tri cua Pi: %f\n",PI);
return 0; }
```

Open terminal:

gcc –E test.c > test.p

Open file test.p and check

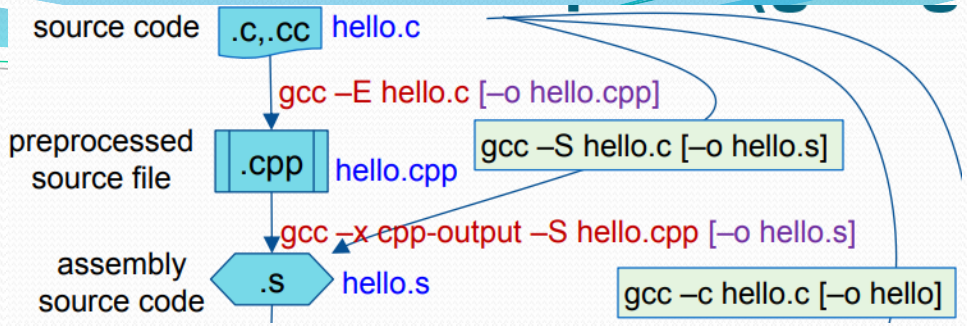
```
extern int ftrylockfile (FILE *__stream) __att

extern void funlockfile (FILE *__stream) __att
# 942 "/usr/include/stdio.h" 3 4

# 2 "testc.c" 2


# 3 "testc.c"
int main()
{
  printf("Hello World\n");
  printf("Gia tri cua Pi: %f\n",3.14159);
  return 0;
}
```

Trình biên dịch GNU C/C++



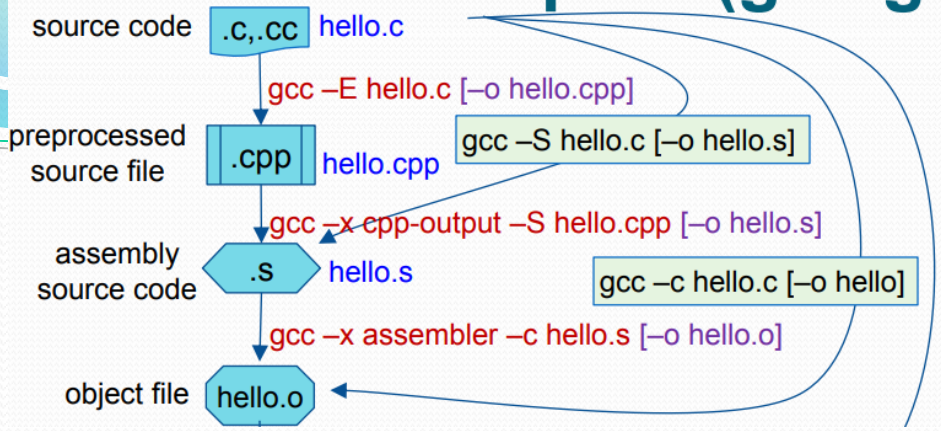
- Pha 2: Compiler, giai đoạn biên dịch ra mã trung gian, là mã assembly và các tùy tối ưu chương trình chủ yếu được thực hiện ở pha này.
  - Kết quả sẽ cho ra file .s, chứa các mã lệnh sau khi biên dịch ngôn ngữ nguồn và không phụ thuộc vào nền máy.
  - Có thể dừng lại ở pha này bằng cách thêm chỉ thị dịch –S
- Ví dụ: tạo file test.s

  Open terminal:

  gcc –S test.c -o test.s

```
        .file   "testc.c"
        .section        .rodata
.LC0:
        .string "Hello World"
.LC2:
        .string "Gia tri cua Pi: %f\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $.LC0, %edi
        call    puts
        movabsq $4614256650576692846, %rax
        movq    %rax, -8(%rbp)
        movsd   -8(%rbp), %xmm0
        movl    $.LC2, %edi
        movl    $1, %eax
        call    printf
        movl    $0, %eax
```

Trình biên dịch GNU C/C++



- Pha 3: Assembler, pha này sẽ biên dịch các mã assembly sang mã máy, tạo thành các file object (.o), file .o này chứa cả cá thông tin về debug và liên kết chương trình
  - File .o là định dạng nhị phân
  - Có thể dừng lại ở pha này bằng cách thêm chỉ thị –c
- Ví dụ: tạo file test.o

  Open terminal:

  gcc –c test.c –o test.o

Trình biên dịch GNU C/C++

- Pha 4 : Linker, kết hợp tất cả các file object và thư viện để trở thành một file thực thi dạng binary, nó gần giống với object file.
- Ví dụ: tạo file thực thi test

Open terminal:

gcc test.c -o test
./test

Trình biên dịch GNU C/C++

Tùy chọn	Công dụng
-o FILE	Chỉ định tên của file output (khi biên dịch thành file thực thi, nếu không có **-o filename** thì tên file mặc định sẽ là **a.out**)
-c	Chỉ biên dịch mà không linking (i.e. chỉ tạo ra object file *.o)
-I*DIRNAME*	Chỉ tên thư mục *DIRNAME* là nơi chứa các file header (.h) mà gcc sẽ tìm trong đó (mặc định gcc sẽ tự tìm ở các thư mục chuẩn /usr/include, ...)
-L*DIRNAME*	Chỉ tên thư mục *DIRNAME* là nơi chứa các thư viện (.a, .so) mà gcc sẽ tìm trong đó (mặc định gcc sẽ tự tìm ở các thư mục chuẩn /usr/lib, ...)
-O [n]	Tối ưu mã thực thi tạo ra (e.g. -O2, -O3, hoặc -O)
-g	Chèn thêm mã phục vụ công việc debug
-E	Chỉ thực hiện bước tiền xử lý (preprocessing) mà không biên dịch
-S	Chỉ dịch sang mã hợp ngữ chứ không linking (i.e. chỉ tạo ra file *.s)
-l**foo**	Link với file thư viện có tên là lib**foo** (e.g. -l**m**, -l**pthread**)
-ansi	Biên dịch theo chuẩn ANSI C/C++ (sẽ cảnh báo nếu code không chuẩn)

Trình biên dịch GNU C/C++

Ví dụ: Xem lại ví dụ về InputArray.h, InputArray.cpp, Main.cpp

- Biên dịch (không link) một file chương trình nguồn C++ đơn lẻ

g++ -c Main.cpp

- Biên dịch (không link) có sử dụng các file *.h (lưu ý đường dẫn chứa file header)
  - -g: kèm thông tin phục vụ debug
  - -O2: có tối ưu mã

g++ -c -g -I./ -O2 InputArray.cpp

- Liên kết (link) nhiều file đối tượng (object files) đã có

g++ InputArray.o Main.o -o Main

./Main

Biên dịch chương trình C/C++

- Lưu ý khi biên dịch trong Linux
  - Dùng g++ nếu chương trình có chứa mã C lẫn C++
  - Dùng gcc nếu chương trình chỉ có mã C
  - File thực thi tạo ra không có đuôi .exe, .dll như môi trường Windows
- Giả sử ứng dụng của bạn gồm nhiều hơn một file source code, (e.g. Main.cpp và InputArray.cpp). Để tạo thành chương trình thực thi, bạn có thể biên dịch trực tiếp bằng một lệnh gcc như sau:

g++ InputArray.cpp Main.cpp -o Main

- Cách làm thủ công như trên sẽ bất tiện và không hiệu quả khi ứng dụng gồm quá nhiều file (khoảng>10 files ???)
  - Khắc phục sử dụng GNU make

Bài tập

1. Viết chương trình tính giai thừa một số nguyên N.

2. Viết chương trình liệt kê các số nguyên tố trong mảng môt chiều (yêu cầu tạo file header và file source chứa các hàm liên quan : *.h, *.cpp, Main.cpp)

3. Viết chương trình nhập vào một mảng các số nguyên dương chưa biết trước số phần từ và đồng thời không nhập số lượng phần tử từ đầu (nhập vào -1 để kết thúc; nhập đến đâu mở rộng mảng đến đó.

   a) Nhập và xuất mảng ra màn hình.

   b) Sắp xếp mảng tăng dần

   c) Xóa các số lẻ ra khỏi mảng.

Next

- Thư viện liên kết tĩnh và động
- Make, Makefile