Lập trình C/Linux

Trình bày: TS. NGÔ BÁ HÙNG
Website: http://sites.google.com/site/nbhung

Ngôn ngữ phát triển PMMNM

- C, C++
- Shell
- Perl,
- PHP,
- Python
- Java,
- C#

Lập trình C/Linux

- Lập trình C
  - Công cụ cần thiết
  - Trình biên dịch gcc
  - Tập tin tiêu đề
  - Tập tin thư viện hàm
- Tiện ích make
  - Tập tin makefile
  - Macro

Lập trình C/Linux

- Lập trình C
  - Công cụ cần thiết
  - Trình biên dịch gcc
  - Tập tin tiêu đề
  - Tập tin thư viện hàm
- Tiện ích make
  - Tập tin makefile
  - Macro

Công cụ cần thiết

- Trình soạn thảo văn bản (text):
  - vi, nano, gedit, emacs, geany, IDE...
- Trình biên dịch:
  - gcc/GNU, cc/Sun, bcc/Borland
  - g++/GNU, CC/Sun
- Thư viện chuẩn của ngôn ngữ C
  - glibc

Biên dịch chương trình đơn giản

- gcc hello.c
  - Tạo ra tập tin thực thi **a.out**
- gcc -o hello hello.c
  - Tạo ra tập tin thực thi **hello**
- gcc -c hello.c
  - Tạo ra tập tin mã đối tượng **hello.o**
- Thực thi
  - ./a.out
  - ./hello

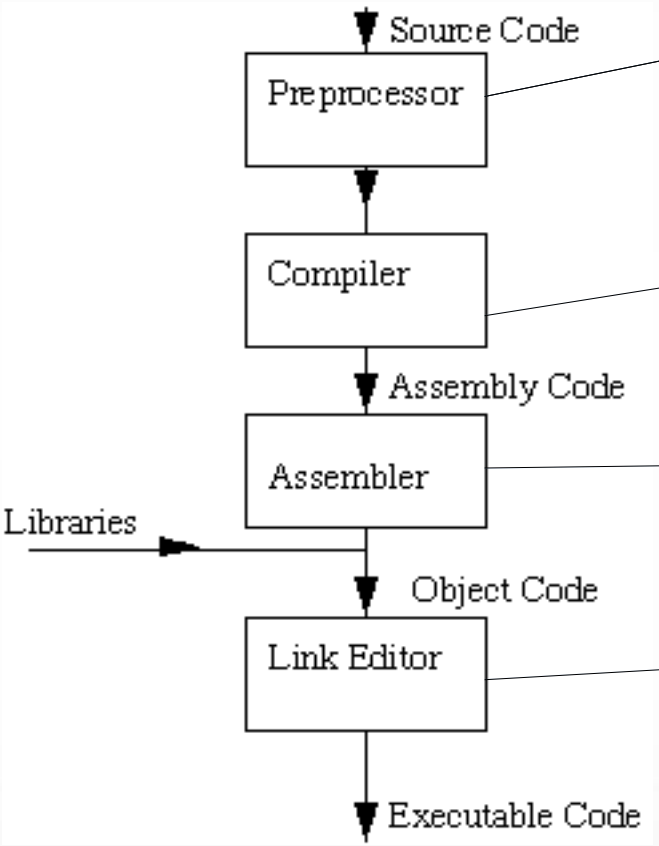
```
/*hello.c*/
#include <stdio.h>
int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Biên dịch chương trình đơn giản

- gcc hello.c
  - Tạo ra tập tin thực thi **a.out**
- gcc -o hello hello.c
  - Tạo ra tập tin thực thi **hello**
- gcc -c hello.c
  - Tạo ra tập tin mã đối tượng **hello.o**
- Thực thi
  - ./a.out
  - ./hello

```
/*hello.c*/
#include <stdio.h>
main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Mô hình biên dịch C



Tiền xử lý mã lệnh: Loại bỏ ghi chú, chèn mã nguồn của các tập tin được include, ...

Biên dịch mã nguồn đã được tiền xử lý thành mã máy

Tạo thành mã đối tượng, có phần mở rộng là .o

Liên kết các hàm được tham khảo lại với nhau để tạo thành chương trình thực thi

Các tùy chọn của gcc

- -Wall: hiển thị toàn bộ các warning
- -ansi: Sử dụng C chuẩn ANSI
- -o: Đặt tên cho tập tin kết quả biên dịch
- -c: Tạo các tập tin đối tượng, không liên kết
- -lm: Liên kết với thư viện toán, nếu trong chương trình có #include math.h

Ví dụ về gcc

- gcc -o hello hello.c
  - Tạo ra tập tin thực thi hello
- gcc -c hello.c bonjour.c chao.c
  - Tạo ra các tập tin hello.o bonjour.o chao.o
- gcc hello.o bonjour.o chao.o -o helloworld
  - Liên kết 3 tập tin mã đối tượng để tạo thành một tập tin thực thi helloworld
  - Tập tin mã đối tượng giúp chỉnh sửa một tập tin không cần biên dịch lại các tập tin khác

Bài tập

1) Viết chương trình giải phương trình bậc 1
2) Viết chương trình tính tổng từ 1 đến n, n nhập từ bàn phím
3) Viết chương trình tính giai thừa n, n nhập từ bàn phím

Tập tin tiêu đề (header file)

- Chứa các **định nghĩa hằng**, **các khai báo về các hàm hệ thống hoặc hàm thư viện** mà một chương trình C có thể gọi sử dụng
- Lưu trữ mặc nhiên ở thư mục chuẩn /usr/include và các thư mục con của thư mục này
- Sử dụng tùy chọn **-I** khi biên dịch để tham khảo đến các tập tin tiêu đề ở một thư mục bất kỳ
  - gcc **-I**/usr/openwin/include myprog.c

Tập tin thư viện hàm

- Chứa các hàm đã được biên dịch trước để có thể được sử dụng lại bởi các chương trình C khác mà không cần phải viết lại
- Các tập tin thư viện hàm chuẩn của hệ thống Linux được lưu trong thư mục **/lib** hoặc **/usr/lib**
- Qui tắc đặt tên:
  - Thư viện tĩnh (static librabry): **lib**Indicat**.a**
  - Thư viện chia sẻ (shared librabry): **lib**Indicat**.so**
  - **libc.a** - Thư viện hàm C; **libm.a** - Thư viện về toán

Sử dụng thư viện hàm

- Mô tả đường dẫn đến tập tin thư viện hàm
  - gcc -o myprog myprog.c **/usr/lib/libm.a**
- Dùng tùy chọn **-l** và **indicat** của thư viện hàm
  - gcc -o myprog myprog.c **-lm**
    - Tìm trong thư mục thư viện hàm chuẩn hệ thống;
    - Sử dụng thư viện chia sẻ **libm.so** trước nếu tồn tại, nếu không sẽ dùng thư viện tĩnh **libm.a**
- Dùng tùy chọn -L để bổ sung thư mục chứa thư viện hàm: gcc -o myprog -L/usr/openwin/lib myprog.c -lX11

  // gcc -o myprog myprog.c /usr/openwin/lib/libX11.so

Xây dựng thư viện hàm tĩnh (1)

```
// File name: hello.c
#include <stdio.h>
void hello(char * name)
{
  printf("Hello %s\n",name);
}
```

```
// File name: bonjour.c
#include <stdio.h>
void bonjour(char *name)
{
  printf("Bonjour %s\n",name);
}
```

```
//File name: mylib.h
void hello(char * name);
void bonjour(char *name);
```

```
// File name: helloworld.c
#include "mylib.h"
int main()
{
  hello("Hung");
  bonjour("Hung");
  return 0;
}
```

```
gcc -c *.c
ls *.o
bonjour.o hello.o helloworld.o
gcc -o helloworld hello.o bonjour.o helloworld.o
./helloworld
Hello Hung
Bonjour Hung
```

Xây dựng thư viện hàm tĩnh (2)

- Tạo tập tin thư viện hàm
  - ar crv libmylib.a hello.o bonjour.o
- Sử dụng thư viện
  - gcc -o helloworld helloworld.o libmylib.a
  - Hoặc gcc -o helloworld helloworld.o -L. -lmylib
- Tiện ích **nm**: xem các hàm sử dụng trong một chương trình, thư viện:
  - nm helloworld
  - nm libmylib.a

Thư viện hàm chia sẻ

- Khắc phục hạn chế của thư viện hàm tĩnh: cùng một hàm nhưng xuất hiện ở nhiều nơi trong bộ nhớ máy tính khi có nhiều tiến trình cùng tham khảo đến hàm làm lãng phí bộ nhớ
- Chương trình sử dụng hàm của thư viện hàm chia sẻ không chứa mã code của hàm mà chứa mã tham khảo đến hàm
- Tiện ích **ldd**: cho biết thư viện chia sẻ nào cần bởi một chương trình

Lập trình C/Linux

- Lập trình C
  - Công cụ cần thiết
  - Trình biên dịch gcc
  - Tập tin tiêu đề
  - Tập tin thư viện hàm
- Tiện ích make
  - Tập tin makefile
  - Macro

Giới thiệu tiện ích make

- Là tiện ích lập trình
- Giúp người lập trình
  - Không phải đánh lại các câu lệnh biên dịch nhiều lần
  - Tránh sai sót khi nhập các tùy chọn biên dịch từ bàn phím
  - Tiết kiệm thời gian biên dịch chương trình vì không biên dịch lại các tập tin nguồn không có sửa đổi
  - Dễ dàng phân phối phần mềm dưới dạng mã nguồn để người cài đặt biên dịch lại khi cài đặt hệ thống

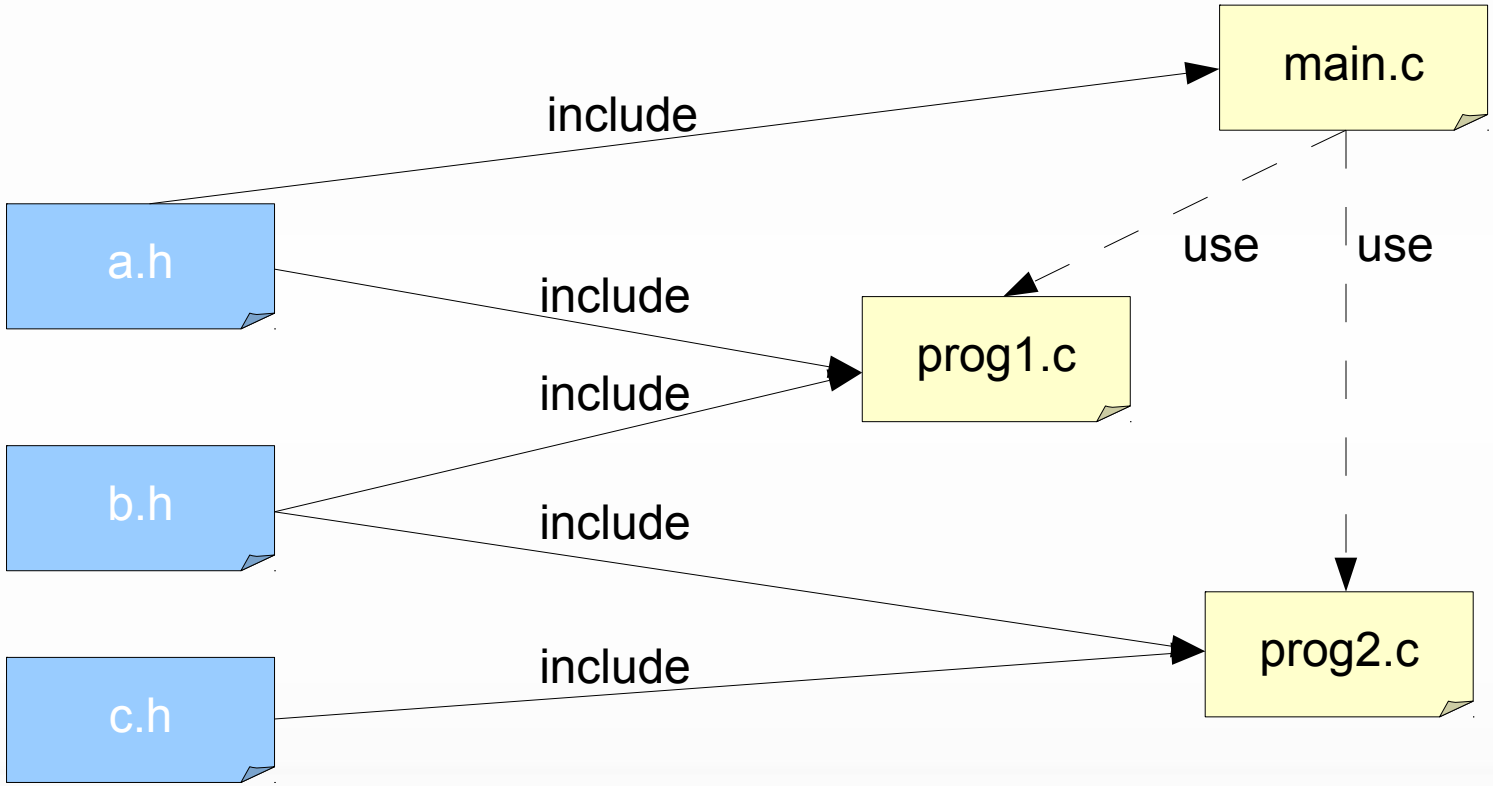
Tập tin mô tả

- Có tên mặc nhiên là **makefile/Makefile**
- Được dùng để chỉ dẫn make cách thức biên dịch/biên dịch lại một cách tự động một chương trình; bao gồm:
  - Các mục tiêu (targets): thường là các tập tin thực thi hoặc các tập tin mã đối tượng cần tạo ra
  - Những sự phụ thuộc (dependencies) để chỉ ra sự phụ thuộc của một mục tiêu vào các tập tin khác
  - Các luật (rules) để chỉ ra cách thức tạo ra các mục tiêu

Cách thức make hoạt động

- make bắt đầu từ một mục tiêu được yêu cầu trong tập tin mô tả Makefile
- Kiểm tra xem mục tiêu hiện tại có phụ thuộc vào các mục tiêu khác không ? Nếu có đi xuống một các đệ qui các mục tiêu con
- Dịch các tập tin nguồn thành các tập tin đối tượng, sau đó liên kết chúng lại thành tập tin thực thi
- Chỉ dịch lại tập tin nguồn thành tập tin đối tượng khi tập tin nguồn này bị sửa đổi

Mã nguồn của một ứng dụng



Makefile cho ứng dụng

```
myapp: main.o prog1.o prog2.o
	gcc -o myapp main.o prog1.o prog2.o
main.o: main.c a.h
	gcc -c main.c
prog1.o: prog1.c a.h b.h
	gcc -c prog1.c
prog2.o: prog2.c b.h c.h
	gcc -c prog2.c
```

Các mục tiêu trong Makefile

```
myapp_t: main.t prog1.t prog2.t
    gcc -o myapp main.o prog1.o prog2.o
main.t: main.c a.h
    gcc -c main.c
prog1.t: prog1.c a.h b.h
    gcc -c prog1.c
prog2.t: prog2.c b.h c.h
    gcc -c prog2.c
```

Những sự phụ thuộc trong Makefile

```
myapp: main.t prog1.t prog2.t
    gcc -o myapp main.o prog1.o prog2.o
main.t: main.c a.h
    gcc -c main.c
prog1.t: prog1.c a.h b.h
    gcc -c prog1.c
prog2.t: prog2.c b.h c.h
    gcc -c prog2.c
```

Các luật trong Makefile

```
myapp: main.t prog1.t prog2.t
    gcc -o myapp main.o prog1.o prog2.o
main.t: main.c a.h
    gcc -c main.c
prog1.t: prog1.c a.h b.h
    gcc -c prog1.c
prog2.t: prog2.c b.h c.h
    gcc -c prog2.c
```

Cú pháp của make

- make
  - Sử dụng tập tin makefile hoặc Makefile trong thư mục hiện hành như tập tin mô tả
  - Tạo mục tiêu đầu tiên trong tập tin mô tả
- make -f MyMakeFile
  - Sử dụng tập tin MyMakeFile như tập tin mô tả
- make target-name
  - Tạo mục tiêu target-name trong tập tin mô tả
  - Mục tiêu all thường được định nghĩa để bao gồm tất cả các mục tiêu

Macro trong makefile

- Macro cho phép viết makefile một cách tổng quát và mềm dẽo hơn, tương tự như việc sử dụng biến và hằng trong lập trình
  - Có nhiều tùy chọn cho việc biên dịch chương trình: phiên bản debug, phiên bản phát hành
  - Thay đổi trình biên dịch tùy thuộc vào hệ thống
- Đinh nghĩa macro: MACRONAME=Value
- Truy cập giá trị: $(MACRONAME), ${MACRONAME} hoặc $MACRONAME

Ví dụ makefile có sử dụng ma

```
all: myapp
# Which compiler
CC = gcc
# Where are include files kept
INCLUDE = .
# Options for development
CFLAGS = -g -Wall -ansi
# Options for release
# CFLAGS = -O -Wall -ansi
myapp: main.o prog1.o prog2.o
   $(CC) -o myapp main.o prog1.o prog2.o
main.o: main.c a.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
prog1.o: prog1.c a.h b.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c prog1.c
prog2.o: prog2.c b.h c.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c prog2.c
```

Macro định sẵn thông dụng

- **$?**: Danh sách các tập tin phụ thuộc có sửa đổi gần đây hơn so với mục tiêu hiện hành
- **$@**: Tên của mục tiêu hiện hành
- **$<**: Tên của tập tin phụ thuộc hiện hành
- **$***: Tên của tập tin phụ thuộc hiền hành không có phần mở rộng
- **-cmd**: Bỏ qua lỗi khi thực thi **cmd**
- **@cmd**: yêu cầu make không in **cmd** ra màn hình trước khi thực thi nó

Định nghĩa nhiều mục tiêu

```
all: myapp
CC = gcc
# Where to install
INSTDIR = /usr/local/bin
# Where are include files kept
INCLUDE = .
# Options for development
CFLAGS = -g -Wall -ansi
# Options for release
# CFLAGS = -O -Wall -ansi
myapp: main.o prog1.o prog2.o
   $(CC) -o myapp main.o prog1.o prog2.o
main.o: main.c a.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
prog1.o: prog1.c a.h b.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c prog1.c
prog2.o: prog2.c b.h c.h
   $(CC) -I$(INCLUDE) $(CFLAGS) -c prog2.c
```

```
clean:
   -rm main.o prog1.o prog2.o
install: myapp
   @if [ -d $(INSTDIR) ]; \
     then \
     cp myapp $(INSTDIR);\
     chmod a+x $(INSTDIR)/myapp;\
     chmod og-w $(INSTDIR)/myapp;\
     echo “Installed in $(INSTDIR)”;\
   else \
     echo “Sorry, $(INSTDIR) does not exist”;\
   fi
```

Thực thi make với nhiều mục tiêu

```
$ rm *.o myapp
$ make -f MyMakeFile
gcc -I. -g -Wall -ansi -c main.c
gcc -I. -g -Wall -ansi -c prog1.c
gcc -I. -g -Wall -ansi -c prog2.c
gcc -o myapp main.o prog1.o prog2.o
$ make -f MyMakeFile
make: Nothing to be done for ‘all’.
```

```
$ rm myapp
$ make -f MyMakeFile install
gcc -o myapp main.o prog1.o prog2.o
Installed in /usr/local/bin
$ make -f MyMakeFile clean
rm main.o prog1.o prog2.o
$
```

Những vấn đề khác về make

- Built-in rule
  - make -p
- Suffix and Pattern Rules
- Managing Libraries with make
- Makefiles and Subdirectories

Bài tập

- Truy cập đến sourceforge.net / code.google.com
- Tìm và download mã nguồn của một dự án phát triển bằng ngôn ngữ C
- Tìm tập tin makefile trong thư mục dự án, cho biết có các mục tiêu nào
- Chuyển vào thư mục của dự án
- Gọi tiện ích make với các mục tiêu khác nhau: tạo ra ứng dụng, cài đặt ứng dụng
- Chạy thử ứng dụng vừa tạo ra/cài đặt