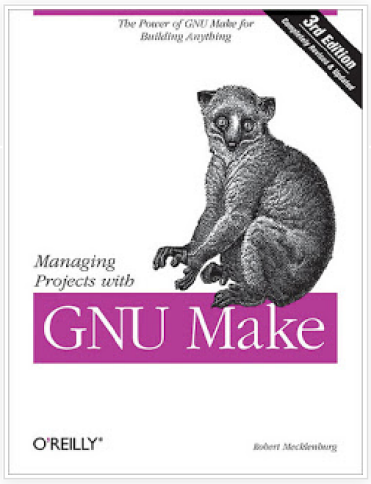
Makefile (Part 2)

Thursday, June 25, 2015



Bài viết liên quan:

+ Makefile (Part 1)

3. Khuôn dạng đầy đủ của Makefile

sum.h (giữ nguyên)

sum.c (giữ nguyên)

main.c (thêm #ifdef ... #endif)

```
#include <stdio.h>
#include "sum.h"

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

     int x;
     x= sum(1, 2);

#ifdef DEBUG
     printf("x = %d \n", x);
#endif

     return 1;
}
```

Makefile

```
.PHONY: all, install, clean

TARGET=sum
```

Labels

```
HDRS+= sum.h
CSRCS+= main.c sum.c
CPPSRCS+=

OBJSDIR=./build
OBJS:= $(patsubst %.cpp, $(OBJSDIR)/%.o, $(CPPSRCS))
OBJS+= $(patsubst %.c, $(OBJSDIR)/%.o, $(CSRCS))

CFLAGS += -I./include -DDEBUG -Wall -g
LDFLAGS += -L./lib -lm

CC:= gcc
CXX:= g++

all: ${TARGET}
${TARGET} : $(OBJS)
     @echo " [LINK] $@"
     @mkdir -p $(shell dirname $@)
     @$(CXX) $(OBJS) -o $@ $(LDFLAGS)

$(OBJSDIR)/%.o: %.c $(HDRS)
     @echo " [CC]  $@"
     @mkdir -p $(shell dirname $@)
     @$(CC) -c $< -o $@ $(CFLAGS)

$(OBJSDIR)/%.o: %.cpp $(HDRS)
     @echo " [CXX] $@"
     @mkdir -p $(shell dirname $@)
     @$(CXX) -c $< -o $@ $(CFLAGS)

install:
     cp -rf ${TARGET} /usr/local/bin

clean:
     rm -rf ${OBJSDIR}/*.o
     rm -rf ${TARGET}
```

Giải thích:

3.1 all, install, clean, .PHONY

+ **all**

Là target mặc định, khi bạn thực hiện lệnh **make** thì chương trình make sẽ tiến hành compile để tạo ra chương trình cuối cùng tương ứng với target này, trong ví dụ này là ${TARGET} tương ứng với sum; hai lệnh dưới đây là tương đương

```
make
make all
```

+ **install, clean**

Các target do người dùng tự định nghĩa, tương ứng với các câu lệnh:

```
make install  #thực hiện action cp
make clean    #thực hiện action rm
```

+ **.PHONY target**

Target này có tác dụng chống xung đột cho chương trình **make** trong trường hợp trong thư mục source code có một file source nào đó trùng tên với các target trong Makefile, ví dụ all.c, install.c hay clean.c. Tốt nhất trong Makefile luôn dùng từ khóa này.

References

http://www.makelinux.net/books/
Embedded247
http://linux.die.net/
http://www.tutorialspoint.com/unix
http://www.cprogramming.com/
http://www.tutorialspoint.com/cprogramming
http://www.tutorialspoint.com/cplusplus
http://www.cplusplus.com/

3.2 Phép gán: TARGET, CC, CXX, ...

Các từ in hoa như trên biểu diễn cho phép gán, có thể hiểu nó tương tự như #define trong C, nó giúp cho chương trình sáng sủa, dễ hiểu và rút ngắn Makefile đáng kể.
Phép gán được thực hiện bằng các cách sau:
+ "**=**" gán cố định
+ "**:=**" tương tự như "="
+ "**+=**" phép gán nối dài (append), ví dụ khi viết hai dòng

```
CSRCS+= main.c
CSRCS+= sum.c
=> CSRCS = main.c sum.c
```

Để lấy giá trị của phép gán thì dùng các cách sau:

```
${TARGET}
hoặc
$(CFLAGS)
```

+ HDRS / CSRCS /CPPSRCS

Header , C source, C++ source file. Trong vd trên các file đều nằm trong folder gốc, nếu file nào nằm ở các thư mục con, bạn cũng cần chỉ đúng đường dẫn đến file, vd:

```
 HDRS+= ./src/sum.h
 CSRCS+= main.c \
         ./src/sum.c
 CPPSRCS+=
```

"\" là ký tự xuống dòng.

+ OBJSDIR / OBJS

Thư mục chứa các object / các object

+ CFLAGS / LDFLASG

Các flags chứa các options để pass vào cho compiler, xem lại tại đây (mục 7).
Chú ý -DDEBUG tương ứng với define DEBUG để pass vào source code, ví dụ trong main.c ở trên.

+ CC / CXX

Compiler để compile, trong vd này sử dụng gcc/g++ để chạy trên máy tính chứ không phải board nhúng, khi cross compiler thì bạn chỉ cần đặt đúng compiler tương ứng với platform cho board là được, vd:

```
CC=arm-linux-gcc
CXX=arm-linux-g++
```

3.3 Lệnh Shell

Lệnh Shell được đưa vào trong cặp khóa **$( )** như dưới đây:

```
OBJS:= $(patsubst %.cpp, $(OBJSDIR)/%.o, $(CPPSRCS))
mkdir -p $(shell dirname $@)
```

3.4 Makefile tương đương: $<, $@, $*, $?, ...

Macro	Definition

$<	Source code hiện tại để compile, tương ứng với %.c và %.cpp, ví dụ: main.c, sum.c
$@	Target hiện tại tương ứng với $(OBJSDIR)/%.o, ví dụ: main.o, sum.o
$*	Tương tự $< nhưng không có suffix, ví dụ: main, sum
$?	Danh sách dependency tương ứng với %.c $(HDRS), ví dụ: tương ứng với main.o là main.c và sum.h

Được sử dụng trong đoạn Makefile chính:

```
${TARGET} : $(OBJS)
   @echo " [LINK] $@"
   @mkdir -p $(shell dirname $@)
   @$(CXX) $(OBJS) -o $@ $(LDFLAGS)

 $(OBJSDIR)/%.o: %.c $(HDRS)
   @echo " [CC] $@"
   @mkdir -p $(shell dirname $@)
   @$(CC) -c $< -o $@ $(CFLAGS)

 $(OBJSDIR)/%.o: %.cpp $(HDRS)
   @echo " [CXX] $@"
   @mkdir -p $(shell dirname $@)
   @$(CXX) -c $< -o $@ $(CFLAGS)
```

Bạn đọc có thể trace các macro bằng các dòng lệnh sau:

```
@echo " [CC]  $@ $< $* $?"
@echo " [CXX]  $@ $< $* $?"
```

Nguyên lý hoạt động:

Chỉ cần đoạn lệnh ngắn gọn trên là đủ để để compile tất cả các target được liệt kê trong OBJS thông qua ký tự đại diện "%"; compile main.o từ main.c, compile sum.o từ sum.c, tất nhiên có các dependency HDRS

3.5 Các ký tự khác

+ @ ở đầu các Action trong Makefile

Loại bỏ các dòng trace mặc định của chương trình make khi đang compile

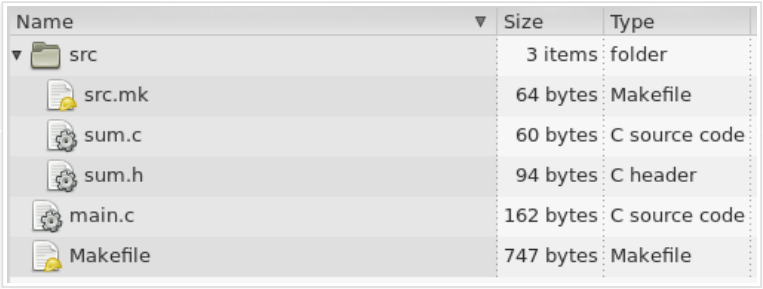
+ -f

Nếu bạn đặt tên của script không phải mặc định là Makefile (ví dụ: MyMakefile) thì bạn cần thêm option -f vào

```
make -f MyMakefile
make -f MyMakefile install
make -f MyMakefile clean
...
```

4. Tạo Makefile theo kiểu module cho project lớn

Nếu project lớn có nhiều subfolder thì bạn nên chia thành nhiều phần cho dễ quản lý, cũng theo vd trên nhưng bây giờ giả sử sum.h và sum.c ở trong một subfolder gọi là src



Name	Size	Type
src	3 items	folder
src.mk	64 bytes	Makefile
sum.c	60 bytes	C source code
sum.h	94 bytes	C header
main.c	162 bytes	C source code
Makefile	747 bytes	Makefile

Trong trường hợp này bạn thêm một makefile phụ là src.mk đặt trong thư mục src

src.mk

```
CSRCS+= ./src/sum.c

CPPSRCS+=

HDRS+= $(wildcard ./src/*.h)
```

Makefile

```
.PHONY: all, install, clean

TARGET:=sum

-include ./src/src.mk
CSRCS+= main.c

OBJSDIR=./build
OBJS:= $(patsubst %.cpp, $(OBJSDIR)/%.o, $(CPPSRCS))
OBJS+= $(patsubst %.c, $(OBJSDIR)/%.o, $(CSRCS))

INCDIR+= -I./src
CFLAGS += -DDEBUG -Wall -g
LDFLAGS += -L./lib -lm



CC:= gcc
CXX:= g++

all: ${TARGET}
${TARGET} : $(OBJS)
	@echo " [LINK] $@"
	@mkdir -p $(shell dirname $@)
	@$(CXX) $(OBJS) -o $@ $(LDFLAGS)

$(OBJSDIR)/%.o: %.c $(HDRS)
	@echo " [CC]  $@"
	@mkdir -p $(shell dirname $@)
	@$(CC) -c $< -o $@ $(CFLAGS) ${INCDIR}

$(OBJSDIR)/%.o: %.cpp $(HDRS)
	@echo " [CXX] $@"
	@mkdir -p $(shell dirname $@)
	@$(CXX) -c $< -o $@ $(CFLAGS) ${INCDIR}

install:
	cp -rf ${TARGET} /usr/local/bin

clean:
	rm -rf ${OBJSDIR}
	rm -rf ${TARGET}
```

Share This

4 bình luận Sắp xếp theo Cũ nhất

Thêm bình luận...

Lê Quỳnh
good
Thích · Phản hồi · 3 năm

Lê Huy Hùng
rất hay, nhiệt liệt hoan nghênh tác giả.
Thích · Phản hồi · 2 · 2 năm

Hao Su
bài trình bày dễ hiểu
Thích · Phản hồi · 2 năm

Nguyễn Ngọc Hùng
this essay is just for people who had understood that problem. For newbies, all you guys should execut more exercises, to understand this problem
Thích · Phản hồi · 1 năm

Plugin bình luận trên Facebook

{ 5 nhận xét... read them below or Comment }

Dieu Tran Minh November 11, 2015 at 11:43 PM

This comment has been removed by the author.

Reply

Anonymous November 11, 2015 at 11:44 PM

There is a mistake in section 3.3, the brackets { } are not used in the example.

Reply

Vu Hai Long September 16, 2018 at 7:54 PM

This comment has been removed by the author.

Reply

Vu Hai Long

like ad September 16, 2018 at 7:54 PM

Reply

Unknown December 10, 2019 at 6:37 PM

I found Long at here ^^
this article is very specific

Reply

Enter your comment...

Comment as: huongpv@gma ▼ Sign out

Publish Preview Notify me

PREV NEXT