

# Makefile进阶

## 开始准备

template.c

代码块

```
1  #include <stdio.h>
2  void fun()
3  {
4      printf("hello fun!\n");
5  }
```

## 构建单目录多文件

mkcode.sh

代码块

```
1  #!/ bin / bash
2  num = 100 suffix = .c template = template.txt
3
4      function
5      mkcode()
6  {
7      cnt = 0 while[$cnt - le $num] do sed "s/fun/fun${cnt}/" ${template} >
8      code${cnt} ${suffix};
9      let cnt++;
10     done
11 }
12 function mkmain()
13 {
14     cnt = 0 > main.c printf "#include <stdio.h>\n\nint main()\n{\n" >> main.c
15     while[$cnt - le $num] do printf " fun${cnt}();\n" let cnt++ done >> main.c echo
16     "}" >> main.c
17 }
18 mkcode mkmain
```

## 构建源文件

代码块

```
1 $ cat code99.c
2 #include <stdio.h>
3     void fun99()
4 {
5     printf("hello world!\n");
6 }
7 $ cat main.c
8 #include <stdio.h>
9     int main()
10 {
11     fun0();
12     fun1();
13     fun2();
14     fun3();
15     fun4();
16     ...
17 }
```

## 场景1：构建单模块，多文件项目

### Step 1: 显示当前目录下的源文件，并把他们的.c全部替换成为.o

#### Makefile

代码块

```
1 $ cat Makefile
2 #获取所有.c
3     SRCS = $(wildcard *.c)
4 #转换.c变成.o
5     OBJS = $(SRC :.c =.o)
6 #查看一下
7     test : @echo "Src:"
8             @echo $(SRCS)
9             @echo "Obj:"
10            @echo $(OBJS)
```

#### 要点:

- SRCS 使用了 wildcard 函数来自动获取当前目录下所有的 .c 文件
- 当你在 Makefile 中使用 wildcard 函数时，它会返回所有匹配给定模式的文件名列表，这些文件名是当前目录（或指定目录，如果函数参数中包含了路径）下的实际存在的文件。这

对于动态地确定需要编译的源文件列表特别有用，尤其是当源文件可能会随着项目的发展而增加或减少时。

- \$(wildcard PATTERN...)

## 查看

代码块

```
1  $ make
2  Src:
3  code63.c code88.c code75.c code72.c code30.c code34.c code90.c code13.c
4  code4.c code53.c code76.c code6.c code99.c code80.c code11.c code100.c
5  code84.c code3.c code23.c code57.c code31.c code28.c code62.c code51.c
6  code26.c code95.c code49.c code16.c code55.c code18.c code65.c code67.c
7  code8.c code22.c code19.c code21.c code47.c code32.c code73.c code74.c
8  code69.c code15.c code87.c code36.c code92.c code44.c code56.c code71.c
9  code14.c code89.c code43.c code50.c code25.c code58.c code45.c code9.c
10 code33.c code1.c code64.c code93.c code98.c code20.c code66.c code83.c
11 code10.c code39.c code41.c code70.c code24.c code17.c code60.c code59.c
12 code61.c code27.c code86.c code37.c code85.c code54.c code5.c code46.c
13 code48.c code29.c code94.c code12.c code96.c code77.c code35.c code7.c
14 code40.c code42.c code97.c code68.c code78.c code52.c code91.c code79.c
15 code81.c code2.c main.c code82.c code38.c code0.c
16 ObjS:
17 code63.o code88.o code75.o code72.o code30.o code34.o code90.o code13.o
18 code4.o code53.o code76.o code6.o code99.o code80.o code11.o code100.o
19 code84.o code3.o code23.o code57.o code31.o code28.o code62.o code51.o
20 code26.o code95.o code49.o code16.o code55.o code18.o code65.o code67.o
21 code8.o code22.o code19.o code21.o code47.o code32.o code73.o code74.o
22 code69.o code15.o code87.o code36.o code92.o code44.o code56.o code71.o
23 code14.o code89.o code43.o code50.o code25.o code58.o code45.o code9.o
24 code33.o code1.o code64.o code93.o code98.o code20.o code66.o code83.o
25 code10.o code39.o code41.o code70.o code24.o code17.o code60.o code59.o
26 code61.o code27.o code86.o code37.o code85.o code54.o code5.o code46.o
27 code48.o code29.o code94.o code12.o code96.o code77.o code35.o code7.o
28 code40.o code42.o code97.o code68.o code78.o code52.o code91.o code79.o
29 code81.o code2.o main.o code82.o code38.o code0.o
```

## Step 2: 所有源文件，编译成为目标文件

更改 `Makefile`

代码块

```
1  #引入编译器
```

```

2 CC = gcc
3 #定义编译选项
4     FLAGS = -c
5 #获取所有.c
6     SRCS = $(wildcard *.c)
7 #转换.c变成.o
8     OBJS = $(SRCS :.c =.o)
9 #定义目标, 让Makefile进行推导
10     .PHONY : all
11             all : $(OBJS) %
12     .o : %.c @echo "compiling ... $<" @$ (CC) $(FLAGS) $ <
13     -o $ @
14 #清理项目
15     .PHONY : clean
16             clean : @echo "clean ... Project"
17                 @echo "$(OBJS)"
18                 @rm -
19     f $(OBJS)

```

## 查看结果

### 代码块

```

1 $ make
2 compiling ... code51.c
3 compiling ... code33.c
4 compiling ... code35.c
5 compiling ... code18.c
6 compiling ... code19.c
7 compiling ... code38.c
8 compiling ... code40.c
9 compiling ... code77.c
10 compiling ... code9.c
11 compiling ... code65.c
12 compiling ... code42.c
13 ...
14
15 $ make clean
16 clean ... Project
17 code98.o code86.o code97.o code55.o code39.o code81.o code53.o code38.o
18 code76.o code5.o code6.o code90.o code80.o code8.o code3.o code48.o code47.o
19 code54.o code57.o code31.o code28.o code62.o code51.o code22.o code26.o
20 code67.o code95.o code16.o code100.o code18.o code65.o code99.o code42.o
21 code72.o code19.o code32.o code73.o code82.o code78.o code63.o code69.o
22 code71.o code15.o code87.o code36.o code92.o code46.o code56.o code21.o
23 code14.o code89.o code25.o code58.o code45.o code9.o code33.o code23.o

```

```
24 code1.o code64.o code93.o code20.o code66.o code27.o code83.o code10.o
25 code11.o code41.o code70.o code24.o code60.o code94.o code61.o code37.o
26 code59.o code29.o code44.o code7.o code12.o code96.o code4.o code35.o
27 code40.o code77.o code49.o code84.o code17.o code13.o code30.o code68.o
28 code52.o code91.o code79.o code74.o code50.o code2.o main.o code85.o code0.o
29 code43.o code88.o code75.o code34.o
```

## Step 3: 编译成为可执行程序

更改 `Makefile`

代码块

```
1 # 引入编译器
2 CC=gcc
3
4 # 定义编译选项
5 FLAGS=-c
6
7 # 获取所有.c
8 SRCS=$(wildcard *.c)
9
10 # 转换.c变成.o
11 OBJS=$(SRCS:.c=.o)
12
13 # 定义目标可执行
14 BIN=project
15
16 $(BIN):$(OBJS)
17     @echo "Link *.o to $(BIN)"
18     @$ (CC) $^ -o $@
19 %.o:%.c
20     @echo "compiling ... $<"
21     @$ (CC) $(FLAGS) $< -o $@
22
23 # 清理项目
24 .PHONY:clean
25 clean:
26     @echo "clean ... Project"
27     @echo "$(OBJS)"
28     @rm -f $(OBJS)
```

查看结果

代码块

```
1  $ make
2  ...
3  compiling ... code82.c
4  compiling ... code55.c
5  compiling ... code29.c
6  compiling ... code44.c
7  compiling ... code33.c
8  compiling ... code12.c
9  compiling ... code96.c
10 compiling ... code7.c
11 compiling ... code49.c
12 Link *.o to project
13 $ ./project
14 hello fun0
15 hello fun1
16 hello fun2
17 ...
```

## 场景2：临时文件、目标可执行与源代码分离

我们编译的时候，会形成很多的.o临时文件，还有可执行程序(目前是一个),这么多的临时文件，和源文件放在一起，就显得非常不优雅，我们想在make的时候，直接将目标.o和可执行程序分别在不同的目录下，这样后期在清理的之后，只要删除目录，就可以去掉所有的临时文件了

更改 `Makefile`

代码块

```
1  # 引入编译器
2  CC=gcc
3
4  # 添加其他命令
5  RM=rm
6  RM_FLAGS=-rf
7  MKDIR=mkdir
8
9  # 定义编译选项
10 FLAGS=-c
11
12 # 获取所有.c
13 SRCS=$(wildcard *.c)
14
15 # 转换.c变成.o
16 OBJS=$(SRCS:.c=.o)
17
18 # 定义目标可执行
```

```

19  BIN=project
20
21  # 添加新建的目录名称
22  OBJS_DIR=objs
23  BIN_DIR=bin
24
25  # 给最终形成的可执行程序添加路径
26  # 这里必须使用:=,要不然makefile会有递归引用的问题.
27  BIN:=$(addprefix $(BIN_DIR)/, $(BIN)) # bin/project
28
29  # 给形成的临时.o都添加目标路径
30  OBJS:=$(addprefix $(OBJS_DIR)/, $(OBJS)) #objs/XX.o
31
32  # 需要新建的文件夹
33  DIR_LIST=$(OBJS_DIR) $(BIN_DIR)
34  all:$(DIR_LIST) $(BIN)
35  $(DIR_LIST):
36      @$(MKDIR) $@
37  $(BIN):$(OBJS)
38      @echo "Link *.o to $(BIN)"
39      @$ (CC) $^ -o $@
40
41  # 这里要添加路径,保证形成的.o放入指定路径下
42  $(OBJS_DIR)/%.o:%.c
43      @echo "compling ... $<"
44      @$ (CC) $(FLAGS) $< -o $@
45
46  .PHONY:clean
47  clean:
48      @echo "clean ... Project"
49      @$ (RM) $(RM_FLAGS) $(DIR_LIST)

```

## 运行结果

### 代码块

```

1  $ make
2  compling ... code51.c
3  compling ... code35.c
4  compling ... code18.c
5  compling ... code19.c
6  compling ... code38.c
7  ...
8  $ tree bin
9  bin
10 └─ project

```

```
11 $ tree objs
12 objs
13 |— code0.o
14 |— code100.o
15 |— code10.o
16 |— code11.o
17 |— code12.o
18 |— code13.o
19 |— code14.o
20 |— code15.o
21 |— code16.o
22 |— code17.o
23 |— code18.o
24 |— code19.o
25 |— code1.o
26 |— code20.o
27 |— code21.o
28 |— code22.o
29 |— code23.o
30 ...
```

## =和:=

=进行赋值，变量的值是整个makefile中最后被指定的值，在make时，会把整个makefile展开，拉通决

定变量的值

代码块

```
1 a = hello
2 b = $(a) bit
3 a = world
4
5 #a = hello
6 #b := $(a) bit
7 #a = world
8
9 all:
10     @echo $(a)
11     @echo $(b)
12 $ make
13 world
14 world bit
```

”:=”就表示直接赋值，赋予当前位置的值，”=”才是真正意义上的直接赋值

代码块

```
1 a = hello
2 b := $(a) bit
3 a = world
4
5 all:
6     @echo $(a)
7     @echo $(b)
8 $ make
9 world
10 hello bit
```



`$(addprefix prefix, names...)` 是一个函数，用于将指定的前缀添加到一组空格分隔

的文件名中。这个函数通常用于将相同的前缀添加到一组文件名或路径中，非常适合在 Makefile 中进行路径拼接操作。

- prefix: 要添加的前缀。
- namesR: 一组空格分隔的文件名或路径。

## 场景3：构建多模块，单可执行文件项目

代码块

```
1 #!/ bin / bash
2 num = 100 suffix = .c template = template.txt
3
4     function
5     mkcode()
6     {
7         cnt = 1 while[$cnt - le $num] do sed "s/fun/fun${cnt}/" ${template} >
8         code${cnt} ${suffix};
9         let cnt++;
10        done
11    }
12    function mkmain()
13    {
14        cnt = 1 > main.c printf "#include <stdio.h>\n\nint main()\n{\n" >> main.c
15        while[$cnt - le $num] do printf " fun${cnt}();\n" let cnt++ done >> main.c echo
16        "}" >> main.c
17    }
18    function mkmodule()
19    {
```

```

16     i = 0 j = 1 for ((; i < 10; i++)) do((end = j + 10))
17         printf "build Module%d\n" $i
18         mkdir -
19         p Module$i for ((; j < end; j++)) do mv code${j}.c Module$i
20 #mv Module$i / code${j }.c.#恢复文件结构
21         done
22         done
23         mkdir -
24         p main
25         mv main.c main
26     }
27 function clean(){
28     rm -f *.c rm -rf Module * rm -rf main} function Usage(){
29     printf "Usage: $0 [-mkcode/-mkmain/-mkmodule/-clr]\n"} function main()
30 {
31     if
32     [ $# -ne 1];
33     then
34         Usage return fi
35         opt = $1 if[$opt == "-mkcode"];
36     then
37         mkcode
38         elif[$opt == "-mkmain"];
39     then
40         mkmain
41         elif[$opt == "-mkmodule"];
42     then
43         mkmodule
44         elif[$opt == "-clr"];
45     then
46         clean else Usage fi
47 }
48 main $ @

```

## 准备模块

### 代码块

```

1  $ ./mkcode.sh -mkcode
2  $ ./mkcode.sh -mkmain
3  $ ./mkcode.sh -mkmodule
4  build Module0
5  build Module1
6  build Module2
7  build Module3
8  build Module4

```

```
9  build Module5
10 build Module6
11 build Module7
12 build Module8
13 build Module9
14 $ ll
15 total 60
16 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 main
17 -rwxrwxr-x 1 whb whb 1258 Sep 24 13:56 mkcode.sh
18 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module0
19 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module1
20 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module2
21 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module3
22 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module4
23 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module5
24 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module6
25 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module7
26 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module8
27 drwxrwxr-x 2 whb whb 4096 Sep 24 15:13 Module9
28 -rw-rw-r-- 1 whb whb 60 Sep 24 13:06 template.txt
```

## 设计 Makefile

### 代码块

```
1  #main子模块名称
2  MODULES := main
3  #获取其他模块的名称
4  MODULES += $(shell ls -d Module*)
5
6  #形成目标文件的名称
7  BIN := project
8  RMF := rm -f
9
10 #循环获取指定模块下的所有.c源文件
11 SRCS := $(foreach module, $(MODULES), $(wildcard $(module)/*.c))
12
13 # 将所有的.c换成.o
14 OBJS := $(SRCS:.c=.o)
15
16 # 基本命令
17 CC=gcc
18
19 # 编译目标程序
20 $(BIN):$(OBJS)
21     @$ (CC) $^ -o $@
```

```

22     @echo "链接 $^ 形成 $@"
23
24 # 形成目标文件
25 %.o:%.c
26     @$(CC) -c $< -o $@
27     @echo "编译$< 形成 $@"
28
29 # 清理项目
30 .PHONY:clean
31 clean:
32     @$(RMF) $(OBJS) $(BIN)
33     @echo "清理 $(OBJS) $(BIN)"
34
35 # 测试, 查看测试结果
36 .PHONY:test
37 test:
38     @echo $(MODULES)
39     @echo "-----"
40     @echo $(SRCS)
41     @echo "-----"
42     @echo $(OBJS)
43     @echo "-----"

```

## 编译

### 代码块

```

1  $ make
2  编译main/main.c 形成 main/main.o
3  编译Module0/code8.c 形成 Module0/code8.o
4  编译Module0/code3.c 形成 Module0/code3.o
5  编译Module0/code5.c 形成 Module0/code5.o
6  编译Module0/code2.c 形成 Module0/code2.o
7  ...
8  编译Module9/code97.c 形成 Module9/code97.o
9  编译Module9/code93.c 形成 Module9/code93.o
10 编译Module9/code98.c 形成 Module9/code98.o
11 编译Module9/code96.c 形成 Module9/code96.o
12 链接 main/main.o Module0/code8.o Module0/code3.o Module0/code5.o
13  Module0/code2.o Module0/code7.o Module0/code10.o Module0/code4.o
14  Module0/code9.o Module0/code1.o Module0/code6.o Module1/code11.o
15  Module1/code17.o Module1/code12.o Module1/code16.o Module1/code18.o
16  Module1/code15.o Module1/code14.o Module1/code13.o Module1/code19.o
17  Module1/code20.o Module2/code24.o Module2/code27.o Module2/code23.o
18  Module2/code28.o Module2/code29.o Module2/code21.o Module2/code22.o
19  Module2/code30.o Module2/code25.o Module2/code26.o Module3/code39.o

```

```
20 Module3/code37.o Module3/code31.o Module3/code35.o Module3/code40.o
21 Module3/code32.o Module3/code36.o Module3/code34.o Module3/code38.o
22 Module3/code33.o Module4/code41.o Module4/code45.o Module4/code42.o
23 Module4/code48.o Module4/code44.o Module4/code49.o Module4/code43.o
24 Module4/code46.o Module4/code50.o Module4/code47.o Module5/code58.o
25 Module5/code59.o Module5/code57.o Module5/code51.o Module5/code60.o
26 Module5/code52.o Module5/code54.o Module5/code56.o Module5/code53.o
27 Module5/code55.o Module6/code70.o Module6/code61.o Module6/code64.o
28 Module6/code62.o Module6/code67.o Module6/code65.o Module6/code68.o
29 Module6/code63.o Module6/code69.o Module6/code66.o Module7/code73.o
30 Module7/code80.o Module7/code77.o Module7/code78.o Module7/code79.o
31 Module7/code74.o Module7/code71.o Module7/code75.o Module7/code72.o
32 Module7/code76.o Module8/code85.o Module8/code90.o Module8/code86.o
33 Module8/code89.o Module8/code81.o Module8/code87.o Module8/code88.o
34 Module8/code82.o Module8/code84.o Module8/code83.o Module9/code100.o
35 Module9/code95.o Module9/code94.o Module9/code99.o Module9/code91.o
36 Module9/code92.o Module9/code97.o Module9/code93.o Module9/code98.o
37 Module9/code96.o 形成 project
```

## 查看编译结果

代码块

```
1 tree .
2 .
3 |— main
4 | |— main.c
5 | |— main.o
6 |— Makefile
7 |— mkcode.sh
8 |— Module0
9 | |— code10.c
10 | |— code10.o
11 | |— code1.c
12 | |— code1.o
13 | |— code2.c
14 | |— code2.o
15 | |— code3.c
16 | |— code3.o
17 | |— code4.c
18 | |— code4.o
19 ...
```



注意:

可以看到，源文件和obj文件会在同一个目录下，这个就是 %.o:%.c 推导出来的

📌 foreach语法介绍

//TODO

## 场景4：构建多模块，单可执行文件项目，文件分离

改写 `makefile`

代码块

```
1  # main子模块名称
2  MODULES := main
3  # 获取其他模块的名称
4  MODULES += $(shell ls -d Module*)
5
6  # 形成目标文件的名称
7  BIN := project
8  RMF := rm -f
9  MKDIR := mkdir
10 # 循环获取指定模块下的所有.c源文件
11 SRCS := $(foreach module, $(MODULES), $(wildcard $(module)/*.c))
12
13 # 将所有的.c换成.o
14 # OBJJS := $(SRCS:.c=.o)
15
16 # 给OBJJS添加临时目录
17 OBJJS_DIR=objs
18 OBJJS := $(patsubst %.c, $(OBJJS_DIR)/%.o, $(SRCS))
19
20 # 基本命令
21 CC=gcc
22
23 # 编译目标程序
24 $(BIN):$(OBJJS)
25     @$ (CC) $^ -o $@
26     @echo "连接 $^ 形成 $@"
27 # 形成目标文件
28 $(OBJJS_DIR)/%.o:%.c
29     @$ (MKDIR) -p $(dir $@)
30     @$ (CC) -c $< -o $@
31     @echo "编译$< 形成 $@"
32
33 # 清理项目
```

```

34 .PHONY:clean
35 clean:
36     @$ (RMF) $(OBJS) $(BIN)
37     @echo "清理 $(OBJS) $(BIN)"
38
39 # 测试, 查看测试结果
40 .PHONY:test
41 test:
42     @echo $(MODULES)
43     @echo "-----"
44     @echo $(SRCS)
45     @echo "-----"
46     @echo $(OBJS)
47     @echo "-----"

```

## 编译

### 代码块

```

1  $ make
2  编译main/main.c 形成 objs/main/main.o
3  编译Module0/code8.c 形成 objs/Module0/code8.o
4  编译Module0/code3.c 形成 objs/Module0/code3.o
5  编译Module0/code5.c 形成 objs/Module0/code5.o
6  编译Module0/code2.c 形成 objs/Module0/code2.o
7  编译Module0/code7.c 形成 objs/Module0/code7.o
8  编译Module8/code82.c 形成 objs/Module8/code82.o
9  编译Module8/code84.c 形成 objs/Module8/code84.o
10 编译Module8/code83.c 形成 objs/Module8/code83.o
11 编译Module9/code100.c 形成 objs/Module9/code100.o
12 编译Module9/code95.c 形成 objs/Module9/code95.o
13 编译Module9/code94.c 形成 objs/Module9/code94.o
14 编译Module9/code99.c 形成 objs/Module9/code99.o
15 编译Module9/code91.c 形成 objs/Module9/code91.o
16 编译Module9/code92.c 形成 objs/Module9/code92.o
17 编译Module9/code97.c 形成 objs/Module9/code97.o
18 编译Module9/code93.c 形成 objs/Module9/code93.o
19 编译Module9/code98.c 形成 objs/Module9/code98.o
20 编译Module9/code96.c 形成 objs/Module9/code96.o
21 连接 objs/main/main.o objs/Module0/code8.o objs/Module0/code3.o
22  objs/Module0/code5.o objs/Module0/code2.o objs/Module0/code7.o
23  objs/Module0/code10.o objs/Module0/code4.o objs/Module0/code9.o
24  objs/Module0/code1.o objs/Module0/code6.o objs/Module1/code11.o
25  objs/Module1/code17.o objs/Module1/code12.o objs/Module1/code16.o
26  objs/Module1/code18.o objs/Module1/code15.o objs/Module1/code14.o
27  objs/Module1/code13.o objs/Module1/code19.o objs/Module1/code20.o

```

```
28  objs/Module2/code24.o  objs/Module2/code27.o  objs/Module2/code23.o
29  objs/Module2/code28.o  objs/Module2/code29.o  objs/Module2/code21.o
30  objs/Module2/code22.o  objs/Module2/code30.o  objs/Module2/code25.o
31  objs/Module2/code26.o  objs/Module3/code39.o  objs/Module3/code37.o
32  objs/Module3/code31.o  objs/Module3/code35.o  objs/Module3/code40.o
33  objs/Module3/code32.o  objs/Module3/code36.o  objs/Module3/code34.o
34  objs/Module3/code38.o  objs/Module3/code33.o  objs/Module4/code41.o
35  objs/Module4/code45.o  objs/Module4/code42.o  objs/Module4/code48.o
36  objs/Module4/code44.o  objs/Module4/code49.o  objs/Module4/code43.o
37  objs/Module4/code46.o  objs/Module4/code50.o  objs/Module4/code47.o
38  objs/Module5/code58.o  objs/Module5/code59.o  objs/Module5/code57.o
39  objs/Module5/code51.o  objs/Module5/code60.o  objs/Module5/code52.o
40  objs/Module5/code54.o  objs/Module5/code56.o  objs/Module5/code53.o
41  objs/Module5/code55.o  objs/Module6/code70.o  objs/Module6/code61.o
42  objs/Module6/code64.o  objs/Module6/code62.o  objs/Module6/code67.o
43  objs/Module6/code65.o  objs/Module6/code68.o  objs/Module6/code63.o
44  objs/Module6/code69.o  objs/Module6/code66.o  objs/Module7/code73.o
45  objs/Module7/code80.o  objs/Module7/code77.o  objs/Module7/code78.o
46  objs/Module7/code79.o  objs/Module7/code74.o  objs/Module7/code71.o
47  objs/Module7/code75.o  objs/Module7/code72.o  objs/Module7/code76.o
48  objs/Module8/code85.o  objs/Module8/code90.o  objs/Module8/code86.o
49  objs/Module8/code89.o  objs/Module8/code81.o  objs/Module8/code87.o
50  objs/Module8/code88.o  objs/Module8/code82.o  objs/Module8/code84.o
51  objs/Module8/code83.o  objs/Module9/code100.o  objs/Module9/code95.o
52  objs/Module9/code94.o  objs/Module9/code99.o  objs/Module9/code91.o
53  objs/Module9/code92.o  objs/Module9/code97.o  objs/Module9/code93.o
54  objs/Module9/code98.o  objs/Module9/code96.o  形成 project
```

## 运行

代码块

```
1  $ ./project
2  hello fun1
3  hello fun2
4  hello fun3
5  hello fun4
6  hello fun5
7  ...
```

## 查看形成结构

代码块

```
1  $ tree objs
2  objs
```

```

3  |—— main
4  |  |—— main.o
5  |—— Module0
6  |  |—— code10.o
7  |  |—— code1.o
8  |  |—— code2.o
9  |  |—— code3.o
10 |  |—— code4.o
11 |  |—— code5.o
12 |  |—— code6.o
13 |  |—— code7.o
14 |  |—— code8.o
15 |  |—— code9.o
16 |—— Module1
17 |  |—— code11.o
18 |  |—— code12.o
19 |  |—— code13.o
20 |  |—— code14.o
21 |  |—— code15.o
22 |  |—— code16.o
23 |  |—— code17.o
24 |  |—— code18.o
25 |  |—— code19.o
26 |  |—— code20.o
27 |—— Module2
28 |  |—— code21.o
29 |  |—— code22.o
30 |  |—— code23.o
31 |  |—— code24.o
32 |  |—— code25.o
33 |  |—— code26.o
34 |  |—— code27.o
35 |  |—— code28.o
36 |  |—— code29.o
37 ...

```

这样，所有的临时文件都在同一个临时目录下，而且目录结构也一致

## 清理项目

代码块

```

1  $ make clean
2  清理 objs/main/main.o objs/Module0/code8.o objs/Module0/code3.o
3  objs/Module0/code5.o objs/Module0/code2.o objs/Module0/code7.o
4  objs/Module0/code10.o objs/Module0/code4.o objs/Module0/code9.o
5  objs/Module0/code1.o objs/Module0/code6.o objs/Module1/code11.o

```

```
6  objs/Module1/code17.o objs/Module1/code12.o objs/Module1/code16.o
7  objs/Module1/code18.o objs/Module1/code15.o objs/Module1/code14.o
8  objs/Module1/code13.o objs/Module1/code19.o objs/Module1/code20.o
9  objs/Module2/code24.o objs/Module2/code27.o objs/Module2/code23.o
10 objs/Module2/code28.o objs/Module2/code29.o objs/Module2/code21.o
11 objs/Module2/code22.o objs/Module2/code30.o objs/Module2/code25.o
12 objs/Module2/code26.o objs/Module3/code39.o objs/Module3/code37.o
13 objs/Module3/code31.o objs/Module3/code35.o objs/Module3/code40.o
14 objs/Module3/code32.o objs/Module3/code36.o objs/Module3/code34.o
15 objs/Module3/code38.o objs/Module3/code33.o objs/Module4/code41.o
16 objs/Module4/code45.o objs/Module4/code42.o objs/Module4/code48.o
17 objs/Module4/code44.o objs/Module4/code49.o objs/Module4/code43.o
18 objs/Module4/code46.o objs/Module4/code50.o objs/Module4/code47.o
19 objs/Module5/code58.o objs/Module5/code59.o objs/Module5/code57.o
20 objs/Module5/code51.o objs/Module5/code60.o objs/Module5/code52.o
21 objs/Module5/code54.o objs/Module5/code56.o objs/Module5/code53.o
22 objs/Module5/code55.o objs/Module6/code70.o objs/Module6/code61.o
23 objs/Module6/code64.o objs/Module6/code62.o objs/Module6/code67.o
24 objs/Module6/code65.o objs/Module6/code68.o objs/Module6/code63.o
25 objs/Module6/code69.o objs/Module6/code66.o objs/Module7/code73.o
26 objs/Module7/code80.o objs/Module7/code77.o objs/Module7/code78.o
27 objs/Module7/code79.o objs/Module7/code74.o objs/Module7/code71.o
28 objs/Module7/code75.o objs/Module7/code72.o objs/Module7/code76.o
29 objs/Module8/code85.o objs/Module8/code90.o objs/Module8/code86.o
30 objs/Module8/code89.o objs/Module8/code81.o objs/Module8/code87.o
31 objs/Module8/code88.o objs/Module8/code82.o objs/Module8/code84.o
32 objs/Module8/code83.o objs/Module9/code100.o objs/Module9/code95.o
33 objs/Module9/code94.o objs/Module9/code99.o objs/Module9/code91.o
34 objs/Module9/code92.o objs/Module9/code97.o objs/Module9/code93.o
35 objs/Module9/code98.o objs/Module9/code96.o project
```

## 查看清理之后的结构

代码块

```
1  $ tree objs
2  objs
3  ├── main
4  ├── Module0
5  ├── Module1
6  ├── Module2
7  ├── Module3
8  ├── Module4
9  ├── Module5
10 ├── Module6
11 ├── Module7
```

```
12 └─ Module8
13 └─ Module9
```

 patsubst 语法

## 场景5: 引入头文件

### 自动生成头文件

代码块

```
1  #!/ bin / bash
2  num = 100 suffix = .c
3
4      include = .h template = template.txt
5
6      function
7      mkinclude()
8  {
9      cnt = 1 while[$cnt - le $num] do echo "#pragma once" >> code${cnt}
10     ${include} echo "#include<stdio.h>" > code${cnt} ${include} echo "void
11     fun${cnt}();" >> code${cnt} ${include} let cnt++ done
12 }
13
14 function mkcode()
15 {
16     cnt = 1 while[$cnt - le $num] do sed "s/fun/fun${cnt}/" ${template} >
17     code${cnt} ${suffix};
18     let cnt++;
19     done
20 }
21
22 function mkmain()
23 {
24     cnt = 1 > main.c printf "#include <stdio.h>\n" > main.c while[$cnt - le
25     $num] do printf "#include \"code${cnt}.h\"\n" >> main.c
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

28             mv code${j}
29             .h Module$i
30 #mv Module$i / code${j }.c.#恢复文件结构
31             done
32             done
33             mkdir -
34             p main
35             mv main.c main
36 }
37 function clean(){
38     rm - f *.c rm - f *.h rm - rf Module * rm - rf main} function Usage(){
39     printf "Usage: $0 [-mkcode/-mkinclude/-mkmain/-mkmodule/-clr]\n"} function
main()
40 {
41     if
42     [$ # - ne 1];
43     then
44     Usage return fi
45     opt = $1 if[$opt == "-mkcode"];
46     then
47     mkcode
48     elif[$opt == "-mkinclude"];
49     then
50     mkinclude
51     elif[$opt == "-mkmain"];
52     then
53     mkmain
54     elif[$opt == "-mkmodule"];
55     then
56     mkmodule
57     elif[$opt == "-clr"];
58     then
59     clean else Usage fi
60 }
61 main $ @

```

#### 代码块

```

1 $ ./mkcode.sh -mkcode
2 $ ./mkcode.sh -mkinclude
3 $ ./mkcode.sh -mkmain
4 $ ./mkcode.sh -mkmodule

```

## 更改makefile

```

1 # 添加 main子模块名称
2 MODULES := main
3 # 获取其他模块的名称
4 MODULES += $(shell ls -d Module*)
5
6 # 形成目标文件的名称
7 BIN := project
8 RMF := rm -f
9 MKDIR := mkdir
10
11 # 循环获取指定模块下的所有.c源文件
12 SRCS := $(foreach module, $(MODULES), $(wildcard $(module)/*.c))
13 INCLUDES :=$(addprefix -I, $(MODULES))
14
15 # 将所有的.c换成.o
16 # OBJS := $(SRCS:.c=.o)
17
18 # 给OBJS添加临时目录
19 OBJS_DIR=objs
20 OBJS := $(patsubst %.c, $(OBJS_DIR)/%.o, $(SRCS))
21
22 # 基本命令
23 CC=gcc
24
25 # 编译目标程序
26 $(BIN):$(OBJS)
27     @$ (CC) $^ -o $@
28     @echo "连接 $^ 形成 $@"
29
30 # 形成目标文件
31 $(OBJS_DIR)/%.o:%.c
32     @$ (MKDIR) -p $(dir $@)
33     @$ (CC) -c $< -o $@ $(INCLUDES)
34     @echo "编译$< 形成 $@"
35
36 # 清理项目
37 .PHONY:clean
38 clean:
39     @$ (RMF) $(OBJS) $(BIN)
40     @echo "清理 $(OBJS) $(BIN)"
41
42 # 测试, 查看测试结果
43 .PHONY:test
44 test:
45     @echo $(MODULES)
46     @echo "-----"
47     @echo $(SRCS)

```

```
48 @echo "-----"
49 @echo $(OBJJS)
50 @echo "-----"
51 @echo $(INCLUDES)
52 @echo $(INCS)
53 @echo "-----"
```

## 编译

### 代码块

```
1 $ make
2 编译main/main.c 形成 objs/main/main.o
3 编译Module0/code3.c 形成 objs/Module0/code3.o
4 ...
5 编译Module9/code93.c 形成 objs/Module9/code93.o
6 编译Module9/code98.c 形成 objs/Module9/code98.o
7 编译Module9/code96.c 形成 objs/Module9/code96.o
8 连接 objs/main/main.o objs/Module0/code3.o objs/Module0/code5.o
9 objs/Module0/code2.o objs/Module0/code7.o objs/Module0/code10.o
10 objs/Module0/code8.o objs/Module0/code4.o objs/Module0/code9.o
11 objs/Module0/code1.o objs/Module0/code6.o objs/Module1/code11.o
12 objs/Module1/code17.o objs/Module1/code12.o objs/Module1/code16.o
13 objs/Module1/code18.o objs/Module1/code15.o objs/Module1/code14.o
14 objs/Module1/code13.o objs/Module1/code19.o objs/Module1/code20.o
15 objs/Module2/code24.o objs/Module2/code27.o objs/Module2/code23.o
16 objs/Module2/code28.o objs/Module2/code29.o objs/Module2/code21.o
17 objs/Module2/code22.o objs/Module2/code30.o objs/Module2/code25.o
18 objs/Module2/code26.o objs/Module3/code39.o objs/Module3/code37.o
19 objs/Module3/code31.o objs/Module3/code35.o objs/Module3/code40.o
20 objs/Module3/code32.o objs/Module3/code36.o objs/Module3/code34.o
21 objs/Module3/code38.o objs/Module3/code33.o objs/Module4/code41.o
22 objs/Module4/code45.o objs/Module4/code42.o objs/Module4/code48.o
23 objs/Module4/code44.o objs/Module4/code49.o objs/Module4/code43.o
24 objs/Module4/code46.o objs/Module4/code50.o objs/Module4/code47.o
25 objs/Module5/code58.o objs/Module5/code59.o objs/Module5/code57.o
26 objs/Module5/code51.o objs/Module5/code60.o objs/Module5/code52.o
27 objs/Module5/code54.o objs/Module5/code56.o objs/Module5/code53.o
28 objs/Module5/code55.o objs/Module6/code70.o objs/Module6/code61.o
29 objs/Module6/code64.o objs/Module6/code62.o objs/Module6/code67.o
30 objs/Module6/code65.o objs/Module6/code68.o objs/Module6/code63.o
31 objs/Module6/code69.o objs/Module6/code66.o objs/Module7/code73.o
32 objs/Module7/code74.o objs/Module7/code80.o objs/Module7/code77.o
33 objs/Module7/code78.o objs/Module7/code72.o objs/Module7/code71.o
34 objs/Module7/code79.o objs/Module7/code75.o objs/Module7/code76.o
35 objs/Module8/code85.o objs/Module8/code90.o objs/Module8/code86.o
```

```
36 objs/Module8/code89.o objs/Module8/code81.o objs/Module8/code87.o
37 objs/Module8/code88.o objs/Module8/code82.o objs/Module8/code84.o
38 objs/Module8/code83.o objs/Module9/code100.o objs/Module9/code95.o
39 objs/Module9/code94.o objs/Module9/code99.o objs/Module9/code91.o
40 objs/Module9/code92.o objs/Module9/code97.o objs/Module9/code93.o
41 objs/Module9/code98.o objs/Module9/code96.o 形成 project
```

代码块

```
1 $ tree Module0
2 Module0
3 |— code10.c
4 |— code10.h
5 |— code1.c
6 |— code1.h
7 |— code2.c
8 |— code2.h
9 |— code3.c
10 |— code3.h
11 |— code4.c
12 |— code4.h
13 |— code5.c
14 |— code5.h
15 |— code6.c
16 |— code6.h
17 |— code7.c
18 |— code7.h
19 |— code8.c
20 |— code8.h
21 |— code9.c
22 |— code9.h
```

## 场景6：多 Makefile ， 多可执行程序

可以把之前的内容看做项目的一个部分，拷贝形成多个部分，也可以用如下脚本自动形成：

```
mkcode.sh
```

代码块

```
1 $ cat mkcode.sh
2 #!/ bin / bash
3     num = 100 suffix =.c
4         include =.h template = template.txt
5             function mkinclude()
6 {
```

```

7     cnt = 1 while[$cnt - le $num] do echo "#pragma once" >> code${cnt}
      ${include} echo "#include<stdio.h>" > code${cnt} ${include} echo "void
      fun${cnt}();" >> code${cnt} ${include} let cnt++ done
8   }
9   function mkcode()
10  {
11      cnt = 1 while[$cnt - le $num] do sed "s/fun/fun${cnt}/" ${template} >
      code${cnt} ${suffix};
12      let cnt++;
13      done
14  }
15  function mkmain()
16  {
17      cnt = 1 > main.c printf "#include <stdio.h>\n" > main.c while[$cnt - le
      $num] do printf "#include \"code${cnt}.h\"\n" >> main.c
18
      let cnt++;
19      done
20      cnt = 1 printf "int main()\n{\n" >> main.c while[$cnt - le $num] do
      printf " fun${cnt}();\n" let cnt++ done >> main.c echo "}" >> main.c
21  }
22  function mkmodule()
23  {
24      i = 0 j = 1 for ((; i < 10; i++)) do((end = j + 10))
25          printf "build Module%d\n" $i
26          mkdir -
27          p Module$i for ((; j < end; j++)) do mv code${j}.c Module$i
28          mv code${j}
29          .h Module$i
30      #mv Module$i / code${j }.c.#恢复文件结构
31          done
32          done
33          mkdir -
34          p main
35          mv main.c main
36  }
37  #根据模版makefile, 在各个目录下形成makefile
38  function mkfile()
39  {
40      i = 0 for ((; i < 10; i++)) do sed "s/project/project${i}/g" subMakefile >
      Module$i / Makefile
41      #构建每一个模块的测试main.c
42
      echo "#include <stdio.h>" >
43      Module$i / main.c echo "int main() {" >> Module$i / main.c done
44  }
45  function clean(){

```

```

46     rm - f *.c rm - f *.h rm - rf Module * rm - rf main} function Usage(){
47     printf "Usage: $0 [-mkcode/-mkinclude/-mkmain/-clr/-mkfile]\n"}
function main()
48 {
49     if
50         [$ # - ne 1];
51     then
52         Usage return fi
53         opt = $1 if[$opt == "-mkcode"];
54     then
55         mkcode
56         elif[$opt == "-mkinclude"];
57     then
58         mkinclude
59         elif[$opt == "-mkmain"];
60     then
61         mkmain
62         elif[$opt == "-mkmodule"];
63     then
64         mkmodule
65         elif[$opt == "-mkfile"];
66     then
67         mkfile
68         elif[$opt == "-clr"];
69     then
70         clean else Usage fi
71 }
72 main $ @

```

样板 `subMakefile`

代码块

```

1  BIN := project
2  CC := gcc
3  SRCS := $(wildcard *.c)
4  OBJS := $(SRCS:.c=.o)
5
6  $(BIN):$(OBJS)
7      $(CC) -o $@ $^
8  *.o:%.c
9      $(CC) -c $<
10
11 .PHONY:clean
12 clean:
13     rm -f *.o project

```

```
14
15 .PHONY:output
16 output:
17     mkdir -p ../bin
18     mv project ../bin
19     #mv project ../
```

构建好之后，每一个模块都可以独立编译

代码块

```
1  $ tree Module0/
2  Module0/
3  |— code10.c
4  |— code10.h
5  |— code1.c
6  |— code1.h
7  |— code2.c
8  |— code2.h
9  |— code3.c
10 |— code3.h
11 |— code4.c
12 |— code4.h
13 |— code5.c
14 |— code5.h
15 |— code6.c
16 |— code6.h
17 |— code7.c
18 |— code7.h
19 |— code8.c
20 |— code8.h
21 |— code9.c
22 |— code9.h
23 |— main.c
24 └— Makefile
```

整体文件目录结构

代码块

```
1  $ ll
2  total 56
3  -rw-rw-r-- 1 whb whb 426 Oct 8 16:10 Makefile
4  -rwxrwxr-x 1 whb whb 2145 Oct 8 15:57 mkcode.sh
5  drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module0
6  drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module1
```

```
7 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module2
8 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module3
9 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module4
10 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module5
11 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module6
12 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module7
13 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module8
14 drwxrwxr-x 2 whb whb 4096 Oct 8 16:12 Module9
15 -rw-rw-r-- 1 whb whb 240 Oct 8 16:11 subMakefile
16 -rw-rw-r-- 1 whb whb 60 Sep 24 17:06 template.txt
```

## Makefile结构

代码块

```
1 $ find . -name Makefile
2 ./Module0/Makefile
3 ./Module3/Makefile
4 ./Module7/Makefile
5 ./Module8/Makefile
6 ./Module5/Makefile
7 ./Module1/Makefile
8 ./Module2/Makefile
9 ./Module6/Makefile
10 ./Module9/Makefile
11 ./Module4/Makefile
12 ./Makefile
13 $ find . -name main.c
14 ./Module0/main.c
15 ./Module3/main.c
16 ./Module7/main.c
17 ./Module8/main.c
18 ./Module5/main.c
19 ./Module1/main.c
20 ./Module2/main.c
21 ./Module6/main.c
22 ./Module9/main.c
23 ./Module4/main.c
```

## 顶级makefile内容

代码块

```
1 # 获取其他模块的名称
2 MODULES += $(shell ls -d Module*)
3
```

```

4  .PHONY: build
5  build:
6      @for module in $(MODULES); do \
7          (cd $$module && make) || exit 1; \
8          echo "build $$module ... done";\
9      done
10
11 .PHONY: clean
12 clean:
13     @for module in $(MODULES); do \
14         (cd $$module && make clean); \
15         echo "clean $$module ... done";\
16     done
17
18 .PHONY: output
19 output:
20     @for module in $(MODULES); do \
21         (cd $$module && make output); \
22     done

```

## 编译

代码块

```

1  $ make
2  make[1]: Entering directory `/home/whb/mkfile/Module0'
3  gcc -c -o code3.o code3.c
4  gcc -c -o code5.o code5.c
5  gcc -c -o code2.o code2.c
6  gcc -c -o code7.o code7.c
7  gcc -c -o code10.o code10.c
8  gcc -c -o main.o main.c
9  gcc -c -o code8.o code8.c
10 gcc -c -o code4.o code4.c
11 gcc -c -o code9.o code9.c
12 gcc -c -o code1.o code1.c
13 gcc -c -o code6.o code6.c
14 gcc -o project0 code3.o code5.o code2.o code7.o code10.o main.o code8.o
15 code4.o code9.o code1.o code6.o
16 make[1]: Leaving directory `/home/whb/mkfile/Module0'
17 build Module0 ... done
18 ...

```

## 编译结果

```
Module0
 2  |— code10.c
 3  |— code10.h
 4  |— code10.o
 5  |— code1.c
 6  |— code1.h
 7  |— code1.o
 8  |— code2.c
 9  |— code2.h
10  |— code2.o
11  |— code3.c
12  |— code3.h
13  |— code3.o
14  |— code4.c
15  |— code4.h
16  |— code4.o
17  |— code5.c
18  |— code5.h
19  |— code5.o
20  |— code6.c
21  |— code6.h
22  |— code6.o
23  |— code7.c
24  |— code7.h
25  |— code7.o
26  |— code8.c
27  |— code8.h
28  |— code8.o
29  |— code9.c
30  |— code9.h
31  |— code9.o
32  |— main.c
33  |— main.o
34  |— Makefile
35  |— project0
36  ...
```

## 清理

### 代码块

```
1  $ make clean
2  make[1]: Entering directory `/home/whb/mkfile/Module0'
3  rm -f *.o project0
4  make[1]: Leaving directory `/home/whb/mkfile/Module0'
5  clean Module0 ... done
```

```
6 make[1]: Entering directory `/home/whb/mkfile/Module1'
7 rm -f *.o project1
8 make[1]: Leaving directory `/home/whb/mkfile/Module1'
9 clean Module1 ... done
10 make[1]: Entering directory `/home/whb/mkfile/Module2'
11 rm -f *.o project2
12 make[1]: Leaving directory `/home/whb/mkfile/Module2'
13 clean Module2 ... done
14 make[1]: Entering directory `/home/whb/mkfile/Module3'
15 rm -f *.o project3
16 make[1]: Leaving directory `/home/whb/mkfile/Module3'
17 clean Module3 ... done
18 make[1]: Entering directory `/home/whb/mkfile/Module4'
19 ...
```

## 查看清理结果

代码块

```
1 Module0
2 |— code10.c
3 |— code10.h
4 |— code1.c
5 |— code1.h
6 |— code2.c
7 |— code2.h
8 |— code3.c
9 |— code3.h
10 |— code4.c
11 |— code4.h
12 |— code5.c
13 |— code5.h
14 |— code6.c
15 |— code6.h
16 |— code7.c
17 |— code7.h
18 |— code8.c
19 |— code8.h
20 |— code9.c
21 |— code9.h
22 |— main.c
23 |— Makefile
```

## 编译并发布

```
$ make && make output
```

## 查看结果

代码块

```
1  $ ll
2  total 60
3  2drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 bin
4  -rw-rw-r-- 1 whb whb 424 Oct 8 16:20 Makefile
5  -rwxrwxr-x 1 whb whb 2145 Oct 8 15:57 mkcode.sh
6  drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module0
7  drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module1
8  drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module2
9  drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module3
10 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module4
11 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module5
12 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module6
13 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module7
14 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module8
15 drwxrwxr-x 2 whb whb 4096 Oct 8 16:23 Module9
16 -rw-rw-r-- 1 whb whb 240 Oct 8 16:11 subMakefile
17 -rw-rw-r-- 1 whb whb 60 Sep 24 17:06 template.txt
18 $ ll bin
19 total 120
20 -rwxrwxr-x 1 whb whb 9096 Oct 8 16:23 project0
21 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project1
22 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project2
23 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project3
24 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project4
25 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project5
26 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project6
27 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project7
28 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project8
29 -rwxrwxr-x 1 whb whb 9112 Oct 8 16:23 project9
```

形成多可执行程序