41P-makefile 基础规则

makefile： 管理项目。

命名：makefile　Makefile --- make 命令

1 个规则：

目标：依赖条件
（一个 tab 缩进）命令

1. 目标的时间必须晚于依赖条件的时间，否则，更新目标

2. 依赖条件如果不存在，找寻新的规则去产生依赖条件。

ALL：指定 makefile 的终极目标。

2 个函数：

src = $(wildcard ./*.c): 匹配当前工作目录下的所有.c 文件。将文件名组成列表，赋值给变量 src。 src = add.c sub.c div1.c

obj = $(patsubst %.c, %.o, $(src)): 将参数 3 中，包含参数 1 的部分，替换为参数 2。obj = add.o sub.o div1.o

clean: (没有依赖)

-rm -rf $(obj) a.out “-”：作用是，删除不存在文件时，不报错。顺序执行结束。

3 个自动变量：

$@: 在规则的命令中，表示规则中的目标。

$^: 在规则的命令中，表示所有依赖条件。

$<: 在规则的命令中，表示第一个依赖条件。如果将该变量应用在模式规则中，它可将依赖

条件列表中的依赖依次取出，套用模式规则。

模式规则：

```
%.o:%.c
    gcc -c $< -o %@
```

静态模式规则：

```
$(obj):%.o:%.c
    gcc -c $< -o %@
```

伪目标：

```
.PHONY: clean ALL
```

参数：

-n：模拟执行 make、make clean 命令。

-f：指定文件执行 make 命令。　　xxxx.mk

下面来一步一步升级 makefile

第一个版本的 Makefile：

makefile 的依赖是从上至下的，换句话说就是目标文件是第一句里的目标，如果不满足执行依赖，就会继续向下执行。如果满足了生成目标的依赖，就不会再继续向下执行了。

make 会自动寻找规则里需要的材料文件，执行规则下面的行为生成规则中的目标。

```
1
2 hello:hello.o
3     gcc hello.o -o hello
4
5 hello.o:hello.c
6     gcc -c hello.c -o hello.o
```

执行 make 指令

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
hello.c  Makefile
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c hello.c -o hello.o
gcc hello.o -o hello
zhcode@ubuntu:~/Code/Mydir/make$ ls
hello  hello.c  hello.o  Makefile
zhcode@ubuntu:~/Code/Mydir/make$
```

运行 hello，没有问题

```
zhcode@ubuntu:~/Code/Mydir/make$ ./hello
my wife is megumin!!!
hello megumin!!!
my wife is alice!!!
hello alice!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

42P-makefile 的一个规则

修改 hello.c 如下：

```c
#include<stdio.h>

int add(int , int);
int sub(int , int);
int div1(int , int);

int main(int argc, char *argv[]){
    int a = 36, b = 12;

    printf("%d + %d = %d\n", a, b, add(a, b));
    printf("%d - %d = %d\n", a, b, sub(a, b));
    printf("%d / %d = %d\n", a, b, div1(a, b));

    printf("my wife is megumin!!!\n");

    return 0;
}
```

此时要进行编译，则需要多文件联合编译：
gcc hello.c add.c sub.c div1.c -o a.out

```
zhcode@ubuntu:~/Code/Mydir/make$ gcc hello.c add.c sub.c div1.c -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 48
36 - 12 = 24
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

对这个新的代码，写出下面的 makefile

```
a.out:hello.c add.c sub.c div1.c
    gcc hello.c add.c sub.c div1.c -o a.out
```

执行：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  hello.o  makefile  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc hello.c add.c sub.c div1.c -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 48
36 - 12 = 24
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

此时，修改 add.c 为下图

```
1 int add(int a, int b){
2     return a+b+1;
3 }
```

此时，再使用 make，发现了问题

```
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc hello.c add.c sub.c div1.c -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 24
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

可以看到，只修改 add.c，但是编译的时候，其他.c 文件也重新编译了，这不太科学。
明明只改了一个，全部都重新编译了

于是将 makefile 改写如下：

```
1 a.out:hello.o add.o sub.o div1.o
2     gcc hello.o add.o sub.o div1.o -o a.out
3
4 hello.o:hello.c
5     gcc -c hello.c -o hello.o
6
7 add.o:add.c
8     gcc -c add.c -o add.o
9
10 sub.o:sub.c
11     gcc -c sub.c -o sub.o
12
13 div1.o:div1.c
14     gcc -c div1.c -o div1.o
15
```

执行 make 指令如下

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc -c div1.c -o div1.o
gcc hello.o add.o sub.o div1.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  add.o  a.out  div1.c  div1.o  hello  hello.c  hello.o  makefile  mymake  sub.c  sub.o
zhcode@ubuntu:~/Code/Mydir/make$
```

执行程序输出如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 24
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

此时修改 sub 为下图：

```
1 int sub(int a, int b){
2     return a-b+111;
3 }
```

再次 make

```
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c sub.c -o sub.o
gcc hello.o add.o sub.o div1.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$
```

可以看到，只重新编译了修改过的 sub.c 和最终目标

执行程序：

```
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

makefile 检测原理：
修改文件后，文件的修改时间发生变化，会出现目标文件的时间早于作为依赖材料的时间，出现这种情况的文件会重新编译。
修改 sub.c 后，sub.o 的时间就早于 sub.c ，a.out 的时间也早于 sub.o 的时间了，于是重新编译这俩文件了。

关于 makefile 指定目标问题，先修改 makefile 如下：

```
hello.o:hello.c
    gcc -c hello.c -o hello.o

add.o:add.c
    gcc -c add.c -o add.o

sub.o:sub.c
    gcc -c sub.c -o sub.o

div1.o:div1.c
    gcc -c div1.c -o div1.o

a.out:hello.o add.o sub.o div1.o
    gcc hello.o add.o sub.o div1.o -o a.out
```

只是将 a.out 放在了文件末尾
执行 make，如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c hello.c -o hello.o
zhcode@ubuntu:~/Code/Mydir/make$
```

这是因为，makefile 默认第一个目标文件为终极目标，生成就跑路，这时候可以用 ALL 来指定终极目标
指定目标的 makefile

```
ALL:a.out

hello.o:hello.c
    gcc -c hello.c -o hello.o

add.o:add.c
    gcc -c add.c -o add.o

sub.o:sub.c
    gcc -c sub.c -o sub.o

div1.o:div1.c
    gcc -c div1.c -o div1.o

a.out:hello.o add.o sub.o div1.o
    gcc hello.o add.o sub.o div1.o -o a.out
```

执行：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc -c div1.c -o div1.o
gcc hello.o add.o sub.o div1.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

43P-午后回顾

44P-makefile 两个函数和 clean

src = $(wildcard *.c)
找到当前目录下所有后缀为.c 的文件，赋值给 src

obj = $(patsubset %.c,%.o, $(src))
把 src 变量里所有后缀为.c 的文件替换成.o

用这两个函数修改 makefile 如下：

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

ALL:a.out

hello.o:hello.c
    gcc -c hello.c -o hello.o

add.o:add.c
    gcc -c add.c -o add.o

sub.o:sub.c
    gcc -c sub.c -o sub.o

div1.o:div1.c
    gcc -c div1.c -o div1.o

a.out: $(obj)
    gcc $(obj) -o a.out
```

执行，make 指令，如下所示：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc  div1.o  hello.o  add.o  sub.o  -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

每次要删除.o 文件，很恶心，于是改写 makefile 如下：
加了 clean 部分

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

ALL:a.out

hello.o:hello.c
    gcc -c hello.c -o hello.o

add.o:add.c
    gcc -c add.c -o add.o

sub.o:sub.c
    gcc -c sub.c -o sub.o

div1.o:div1.c
    gcc -c div1.c -o div1.o

a.out: $(obj)
    gcc $(obj) -o a.out

clean:
    -rm -rf $(obj) a.out
```

rm 前面的-，代表出错依然执行。比如，待删除文件集合是 5 个，已经手动删除了 1 个，就只剩下 4 个，然而删除命令里面还是 5 个的集合，就会有删除不存在文件的问题，不加这-，就会报错，告诉你有一个文件找不到。加了-就不会因为这个报错。

执行 make：

```
zhcode@ubuntu:~/Code/Mydir/make$ make
make: Nothing to be done for 'ALL'.
zhcode@ubuntu:~/Code/Mydir/make$
```

由于没有文件变动，a.out 的时间戳晚于所有依赖文件，这里 make 就没干活

于是，执行时加新指令，先模拟执行 clean 部分：

```
zhcode@ubuntu:~/Code/Mydir/make$ make clean -n
rm -rf  div1.o  hello.o  add.o  sub.o  a.out
zhcode@ubuntu:~/Code/Mydir/make$
```

可以看到模拟执行后，会删除哪些文件。

确定没有问题，执行

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  add.o  a.out  div1.c  div1.o  hello  hello.c  hello.o  m2  makefile  mymake  sub.c  sub.o
zhcode@ubuntu:~/Code/Mydir/make$ make clean
rm -rf  div1.o  hello.o  add.o  sub.o  a.out
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  m2  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$
```

45P-makefile3 个自动变量和模式规则

3 个自动变量

$@ ：在规则命令中，表示规则中的目标

$< ：在规则命令中，表示规则中的第一个条件，如果将该变量用在模式规则中，它可以将依赖条件列表中的依赖依次取出，套用模式规则

$^ ：在规则命令中，表示规则中的所有条件，组成一个列表，以空格隔开，如果这个列表中有重复项，则去重

用自动变量修改 makefile，如下：

```
1  src = $(wildcard *.c)  # add.c sub.c div1.c hello.c
2
3  obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o
4
5  ALL:a.out
6
7  hello.o:hello.c
8      gcc -c $< -o $@
9
10 add.o:add.c
11     gcc -c $< -o $@
12
13 sub.o:sub.c
14     gcc -c $< -o $@
15
16 div1.o:div1.c
17     gcc -c $< -o $@
18
19 a.out: $(obj)
20     gcc $^ -o $@
21
22 clean:
23     -rm -rf $(obj) a.out
```

sub，add 这些指令中使用$<和$^都能达到效果，但是为了模式规则，所以使用的$<
执行 make，如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  m2  m3  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc div1.o hello.o add.o sub.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

再来，上面这个 makefile，可扩展性不行。比如，要添加一个乘法函数，就需要在 makefile 里面增加乘法函数的部分。不科学，所以，模式规则就来了

```
%.o:%.c
    gcc -c $< -o $@
```

修改 makefile，如下：

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

ALL:a.out

%.o:%.c
    gcc -c $< -o $@

a.out: $(obj)
    gcc $^ -o $@

clean:
    -rm -rf $(obj) a.out
```

执行 make，如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  m2  m3  makefile  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc div1.o hello.o add.o sub.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

这时，增加一个 mul 函数，并添加 mul.c 文件如下：

```c
#include<stdio.h>

int add(int , int);
int sub(int , int);
int div1(int , int);
int mul(int, int);

int main(int argc, char *argv[]){
    int a = 36, b = 12;

    printf("%d + %d = %d\n", a, b, add(a, b));
    printf("%d - %d = %d\n", a, b, sub(a, b));
    printf("%d / %d = %d\n", a, b, div1(a, b));
    printf("%d * %d = %d\n", a, b, mul(a, b));

    printf("my wife is megumin!!!\n");

    return 0;
}
```

mul.c 如下：

```c
int mul(int a, int b){
    return a*b;
}
```

直接执行 make：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  m2  m3  makefile  mul.c  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o
gcc -c mul.c -o mul.o
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc div1.o mul.o hello.o add.o sub.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
36 * 12 = 432
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

增加函数的时候，不用改 makefile，只需要增加.c 文件，改一下源码，就行。很强势。

继续优化 makefile，使用静态模式规则，就是指定模式规则给谁用，这里指定模式规则给 obj 用，以后文件多了，文件集合会有很多个，就需要指定哪个文件集合用什么规则

```
$(obj):%.o:%.c
    gcc -c $< -o $@
```

修改后 makefile 如下：

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

ALL:a.out

$(obj):%.o:%.c
    gcc -c $< -o $@

a.out: $(obj)
    gcc $^ -o $@

clean:
    -rm -rf $(obj) a.out
```

运行如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  div1.c  hello  hello.c  m2  m3  makefile  mul.c  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o
gcc -c mul.c -o mul.o
gcc -c hello.c -o hello.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc div1.o mul.o hello.o add.o sub.o -o a.out
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
36 * 12 = 432
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

再来一个扩展

当前文件夹下有 ALL 文件或者 clean 文件时，会导致 makefile 瘫痪，如下所示，make clean 没有工作

```
zhcode@ubuntu:~/Code/Mydir/make$ touch clean
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  a.out  div1.c  hello    hello.o  m3        mul.c  mymake  sub.o
add.o  clean  div1.o  hello.c  m2       makefile  mul.o  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make clean
make: 'clean' is up to date.
zhcode@ubuntu:~/Code/Mydir/make$
```

用伪目标来解决，添加一行　.PHONY: clean ALL
makefile 如下所示：

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

ALL:a.out

$(obj):%.o:%.c
    gcc -c $< -o $@

a.out: $(obj)
    gcc $^ -o $@

clean:
    -rm -rf $(obj) a.out

.PHONY: clean ALL
```

再来执行 make clean，就不会受到干扰了

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  a.out  div1.c  hello    hello.o  m3        mul.c  mymake  sub.o
add.o  clean  div1.o  hello.c  m2       makefile  mul.o  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make clean
rm -rf  div1.o  mul.o  hello.o  add.o  sub.o  a.out
zhcode@ubuntu:~/Code/Mydir/make$
```

还有一个扩展就是，编译时的参数，-g,-Wall 这些，可以放在 makefile 里面
修改后 makefile 如下：

```
src = $(wildcard *.c)  # add.c sub.c div1.c hello.c

obj = $(patsubst %.c, %.o, $(src)) # add.o sub.o div1.o hello.o

myArgs = -Wall -g
ALL:a.out

$(obj):%.o:%.c
    gcc -c $< -o $@ $(myArgs)

a.out: $(obj)
    gcc $^ -o $@ $(myArgs)

clean:
    -rm -rf $(obj) a.out

.PHONY: clean ALL
```

执行 makefile，如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
add.c  clean  div1.c  hello  hello.c  m2  m3  makefile  mul.c  mymake  sub.c
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c div1.c -o div1.o -Wall -g
gcc -c mul.c -o mul.o -Wall -g
gcc -c hello.c -o hello.o -Wall -g
gcc -c add.c -o add.o -Wall -g
gcc -c sub.c -o sub.o -Wall -g
gcc div1.o mul.o hello.o add.o sub.o -o a.out -Wall -g
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
36 * 12 = 432
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

46P-习题和作业

源码 add.c, sub.c 这些在 src 目录下，.o 文件要放在 obj 目录下，头文件 head.h 在 inc 目录下。如下所示：

```
zhcode@ubuntu:~/Code/Mydir/make$ ls
clean  hello  inc  m2  m3  makefile  mymake  obj  src
zhcode@ubuntu:~/Code/Mydir/make$
```

首先，将 hello.c 中的头文件单独拿出来
如下：

修改 makefile 如下，主要是注意%的匹配理解，只匹配文件名，目录位置要手动添加

```
src = $(wildcard ./src/*.c)  # ./src/add.c ./src/sub.c ...
obj = $(patsubst ./src/%.c, ./obj/%.o, $(src))

inc_path = ./inc

myArgs = -Wall -g

ALL:a.out

$(obj):./obj/%.o:./src/%.c
    gcc -c $< -o $@ $(myArgs) -I $(inc_path)

a.out: $(obj)
    gcc $^ -o $@ $(myArgs)

clean:
    -rm -rf $(obj) a.out

.PHONY: clean ALL
```

执行一波，如下：

```
zhcode@ubuntu:~/Code/Mydir/make$ make
gcc -c src/div1.c -o obj/div1.o -Wall -g -I ./inc
gcc -c src/mul.c -o obj/mul.o -Wall -g -I ./inc
gcc -c src/hello.c -o obj/hello.o -Wall -g -I ./inc
gcc -c src/add.c -o obj/add.o -Wall -g -I ./inc
gcc -c src/sub.c -o obj/sub.o -Wall -g -I ./inc
gcc obj/div1.o obj/mul.o obj/hello.o obj/add.o obj/sub.o -o a.out -Wall -g
zhcode@ubuntu:~/Code/Mydir/make$ ls
a.out  clean  hello  inc  m2  m3  makefile  mymake  obj  src
zhcode@ubuntu:~/Code/Mydir/make$ ./a.out
36 + 12 = 49
36 - 12 = 135
36 / 12 = 3
36 * 12 = 432
my wife is megumin!!!
zhcode@ubuntu:~/Code/Mydir/make$
```

可以说是非常强势了

调用 clean 删除文件，直接用就行

```
zhcode@ubuntu:~/Code/Mydir/make$ make clean
rm -rf  ./obj/div1.o  ./obj/mul.o  ./obj/hello.o  ./obj/add.o  ./obj/sub.o a.out
zhcode@ubuntu:~/Code/Mydir/make$
```

如果 makefile 的名字变化一下，比如，叫 m6

用 m6 执行 makefile，　　make -f m6

用 m6 执行 clean　　make -f m6 clean