

## Makefile 编译多个可执行文件

如果当前文件夹中有多个.c 文件都要生成可执行文件，那该怎么办呢？

将原来的文件重新复制到新建的文件夹 01 中，

```
whd@whd-Lenovo:~/GDBNOTES$ ls
add.c  gdb调试入门.wps  Makefile  makefile(上).wps  sub.h
add.h  main.c               Makefile.1  sub.c
whd@whd-Lenovo:~/GDBNOTES$ mkdir 01
whd@whd-Lenovo:~/GDBNOTES$ ls
01      add.h      main.c      Makefile.1      sub.c
add.c   gdb调试入门.wps  Makefile  makefile(上).wps  sub.h
whd@whd-Lenovo:~/GDBNOTES$ mv *.c 01
whd@whd-Lenovo:~/GDBNOTES$ ls
01      add.h      gdb调试入门.wps  Makefile  Makefile.1  makefile(上).wps  sub.h
whd@whd-Lenovo:~/GDBNOTES$ cd 01
whd@whd-Lenovo:~/GDBNOTES/01$ ls
add.c  main.c  sub.c
whd@whd-Lenovo:~/GDBNOTES/01$ mv *.* 01
mv: 目标"01" 不是目录
whd@whd-Lenovo:~/GDBNOTES/01$ cd
whd@whd-Lenovo:~$ cd GDBNOTES/
whd@whd-Lenovo:~/GDBNOTES$ mv *.* 01
whd@whd-Lenovo:~/GDBNOTES$ ls
01  Makefile
whd@whd-Lenovo:~/GDBNOTES$ cd 01
whd@whd-Lenovo:~/GDBNOTES/01$ ls
add.c  gdb调试入门.wps  Makefile.1      sub.c
add.h  main.c             makefile(上).wps  sub.h
whd@whd-Lenovo:~/GDBNOTES/01$ cd
whd@whd-Lenovo:~$ cd GDBNOTES/
whd@whd-Lenovo:~/GDBNOTES$ mv Makefile 01
whd@whd-Lenovo:~/GDBNOTES$ ls
01
whd@whd-Lenovo:~/GDBNOTES$
```

新建文件夹 02，在该目录下新建 01test.c 和 02test.c。其中 01test.c 和 02test.c 都要生成可执行文件。

```
whd@whd-Lenovo:~/GDBNOTES$ mkdir 02
whd@whd-Lenovo:~/GDBNOTES$ cd 02
whd@whd-Lenovo:~/GDBNOTES/02$ touch 01test.c 02test.c
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c
whd@whd-Lenovo:~/GDBNOTES/02$
```

这两个文件本身都有 main()函数，

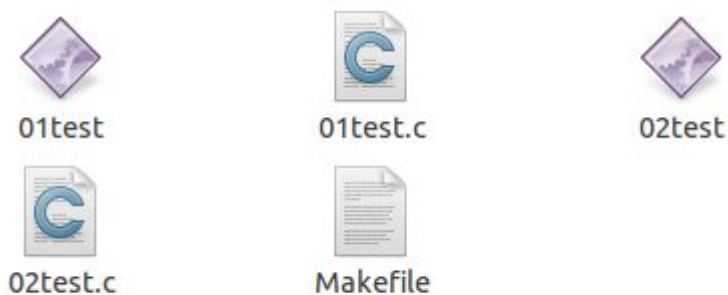
```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
int main(void)
{
    return 0;
}
~
-- 插入 -- 3,12 全部
```

其中 01test.c 要生成 01test,02test.c 要生成 02test。那么我们如何编写 makefile 呢？（[注意](#)：直接 vi Makefile 就能创建该文件）

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean
BIN=01test 02test
all:${BIN}
~
-- 插入 -- 4,1 全部
```

这样我们就直接生成了两个可执行文件：

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ vi 01test.c
whd@whd-Lenovo:~/GDBNOTES/02$ vi 02test.c
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c
whd@whd-Lenovo:~/GDBNOTES/02$ vi Makefile
whd@whd-Lenovo:~/GDBNOTES/02$ make
cc 01test.c -o 01test
cc 02test.c -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 01test.c 02test 02test.c Makefile
whd@whd-Lenovo:~/GDBNOTES/02$
```



下面解释一下，为什么这么简单的命令就可以生成。实际上在该 makefile 中含有隐含的推导规则，我们可以看到 [BIN](#) 这个变量依赖于两个文件：01test,02test.

接下来的依赖规则是：all 这个目标依赖这两个文件，但是低下我们

没有给出如何生成这个目标的命令，因为这个 all 也是个伪目标。可以将 all 加到 clean 的后面：

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
BIN=01test 02test
all:${BIN}
1,1 顶端
```

要生成 all 就要去生成 BIN，紧接着 Makefile 就要生成 BIN 里面的文件，而我们并没有给出 01test 和 02test 的生成过程，编译器会自动推倒将.c 文件生成可执行文件，01test.c 和 02test.c 自动生成 01test 和 02test（同名的.c 文件生成同名的可执行文件）。

增加 make clean 规则：

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ make
cc 01test.c -o 01test
cc 02test.c -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 01test.c 02test 02test.c Makefile
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f 01test 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile
whd@whd-Lenovo:~/GDBNOTES/02$
```

如果我们不想用自带推导的规则怎么办呢？

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
BIN=01test 02test
all:${BIN}
01test:01test.o
    gcc -Wall -g $^ -o 01test
02test:02test.o
    gcc -Wall -g $^ -o 02test
clean:
    rm -f $(BIN)
~
-- 插入 --
1,11 全部
```

注意生成可执行文件就不需要“-c”了。

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$ vi Makefile
whd@whd-Lenovo:~/GDBNOTES/02$ make
cc -c -o 01test.o 01test.c
gcc -Wall -g 01test.o -o 01test
cc -c -o 02test.o 02test.c
gcc -Wall -g 02test.o -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$
```

可以看到生成 01test 和 02test 是按照我们编写的规则生成的，而生成 01test.o 和 02test.o 我们并没有配置它是如何生成的，是系统隐含推导生成的。如果不想通过隐含推导生成，那么我们只有自己给出这个规则，我们可以使用两种规则：最直接的办法，就是按照之前学习到的内容编写规则，如下：

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
BIN=01test 02test
all:${BIN}
01test.o:01test.c
    gcc -Wall -g -c $^ -o 01test.o
02test.o:02test.c
    gcc -Wall -g -c $^ -o 02test.o
01test:01test.o
    gcc -Wall -g $^ -o 01test
02test:02test.o
    gcc -Wall -g $^ -o 02test
clean:
    rm -f *.o ${BIN}
```

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f *.o 01test 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$
```

也可以使用以下两种规则编写，简化内容，两种规则是一样的，只是语法的差别而已：

### (1) 模式规则

`%o.o:%c`



```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
BIN=01test 02test
all:${BIN)
%.o:%.c
    gcc -Wall -g -c $^ -o $@
01test:01test.o
    gcc -Wall -g $^ -o 01test
02test:02test.o
    gcc -Wall -g $^ -o 02test
clean:
    rm -f *.o $(BIN)
~
~
"Makefile" 11L, 191C 1,1 全部
```

就可以将所有的.c 文件生成.o 文件。

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ vi Makefile
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 01test.o 02test.c Makefile makefile编译多个可执行文件.wps
01test.c 02test 02test.o Makefile.1
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f *.o 01test 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile Makefile.1 makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$
```

我们可以看到这次生成的.o 是按照我们编写的规则生成的。

## (2)后缀规则

我们也可以使用后缀规则，将前面的模式规则注释掉,添加后缀规则；  
结果也是一样的：

**.C.O:**

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
BIN=01test 02test
all:${BIN)
#%.o:%.c
#    gcc -Wall -g -c $^ -o $@
.C.O:
    gcc -Wall -g -c $^ -o $@
01test:01test.o
    gcc -Wall -g $^ -o 01test
02test:02test.o
    gcc -Wall -g $^ -o 02test
clean:
    rm -f *.o $(BIN)
~
~
"Makefile" 11L, 191C 1,1 全部
```

结果也是一样的：

```
whd@whd-Lenovo:~/GDBNOTES/02$ vi Makefile
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile Makefile.1 makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 01test.o 02test.c Makefile makefile编译多个可执行文件.wps
01test.c 02test 02test.o Makefile.1
whd@whd-Lenovo:~/GDBNOTES/02$
```

如我们做一些更加专业的 Makefile 的话，我们通常还会定义一些变量，比如说：

```
CC=gcc
CFLAGS=-Wall -g
```

修改后的 Makefile 为：

```
PHONY:clean all
CC=gcc
CFLAGS=-Wall -g
BIN=01test 02test
all:$(BIN)
#%.o:%.c
# $(CC) $(CFLAGS) -c $^ -o $@
.c.o:
$(CC) $(CFLAGS) -c $^ -o $@
01test:01test.o
$(CC) $(CFLAGS) $^ -o 01test
02test:02test.o
$(CC) $(CFLAGS) $^ -o 02test
clean:
rm -f *.o $(BIN)
```

运行结果和前面的一致：

```
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f *.o 01test 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 02test.c Makefile Makefile.1 makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 01test.o 02test.c Makefile makefile编译多个可执行文件.wps
01test.c 02test 02test.o Makefile.1
whd@whd-Lenovo:~/GDBNOTES/02$
```

这样我们就可以生成多个可执行文件，如果我们要增加一个可执行文件 03test，编写 03test.c 的内容如下：

```
int main(void)
{
    return 0;
}
```

1,1 顶端

修改 Makefile 文件的内容：

```
.PHONY:clean all
CC=gcc
CFLAGS=-Wall -g
BIN=01test 02test 03test
all:$(BIN)
#%.o:%.c
# $(CC) $(CFLAGS) -c $^ -o $@
.c.o:
$(CC) $(CFLAGS) -c $^ -o $@
01test:01test.o
$(CC) $(CFLAGS) $^ -o 01test
02test:02test.o
$(CC) $(CFLAGS) $^ -o 02test
03test:03test.o
$(CC) $(CFLAGS) $^ -o 03test
clean:
rm -f *.o $(BIN)
~
~
~
:wq
```

生成结果如下：

```
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f *.o 01test 02test 03test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 03test.c Makefile.1
02test.c Makefile makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
gcc -Wall -g -c 03test.c -o 03test.o
gcc -Wall -g 03test.o -o 03test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 02test 03test Makefile
01test.c 02test.c 03test.c Makefile.1
01test.o 02test.o 03test.o makefile编译多个可执行文件.wps
whd@whd-Lenovo:~/GDBNOTES/02$
```

实际上我们可以将生成.o 的文件注释掉，如下：

```
CC=gcc
CFLAGS=-Wall -g
BIN=01test 02test 03test
all:${BIN}
%.o:%.c
# $(CC) $(CFLAGS) -c $^ -o $@
.c.o:
$(CC) $(CFLAGS) -c $^ -o $@
#01test:01test.o
# $(CC) $(CFLAGS) $^ -o 01test
#02test:02test.o
# $(CC) $(CFLAGS) $^ -o 02test
#03test:03test.o
# $(CC) $(CFLAGS) $^ -o 03test
clean:
rm -f *.o $(BIN)
~
~
-- 插入 -- 16,2 底端
```

系统就会利用隐含推导直接由.c 文件生成.o 文件：

```
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g 01test.c -o 01test
gcc -Wall -g 02test.c -o 02test
gcc -Wall -g 03test.c -o 03test
whd@whd-Lenovo:~/GDBNOTES/02$
```

由于前面加上了下面的编译选项，它的隐含推导就会发生变化，如上图：



```
CC=gcc
CFLAGS=-Wall -g
```

如果 03test 可执行文件不止依赖于 03test.c 呢。比如说它还有一个模块 pub:

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ touch pub.h pub.c
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test    02test    03test    Makefile    makefile编译多个可执行文件.wps  pub.h
01test.c  02test.c  03test.c  Makefile.1  pub.c
whd@whd-Lenovo:~/GDBNOTES/02$
```

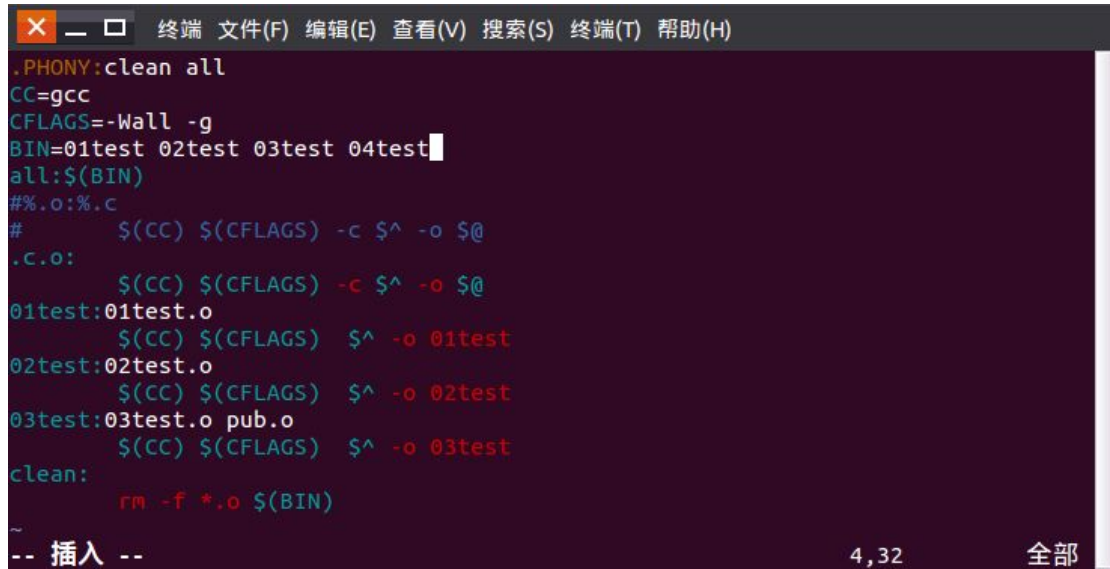
修改 Makefile 文件，添加 pub.o:

```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.PHONY:clean all
CC=gcc
CFLAGS=-Wall -g
BIN=01test 02test 03test
all:${BIN}
#%.o:%.c
#    $(CC) $(CFLAGS) -c ^ -o $@
.c.o:
    $(CC) $(CFLAGS) -c ^ -o $@
01test:01test.o
    $(CC) $(CFLAGS) ^ -o 01test
02test:02test.o
    $(CC) $(CFLAGS) ^ -o 02test
03test:03test.o pub.o
    $(CC) $(CFLAGS) ^ -o 03test
clean:
    rm -f *.o $(BIN)
~
-- 插入 --
1,17 全部
```

就可以生成:

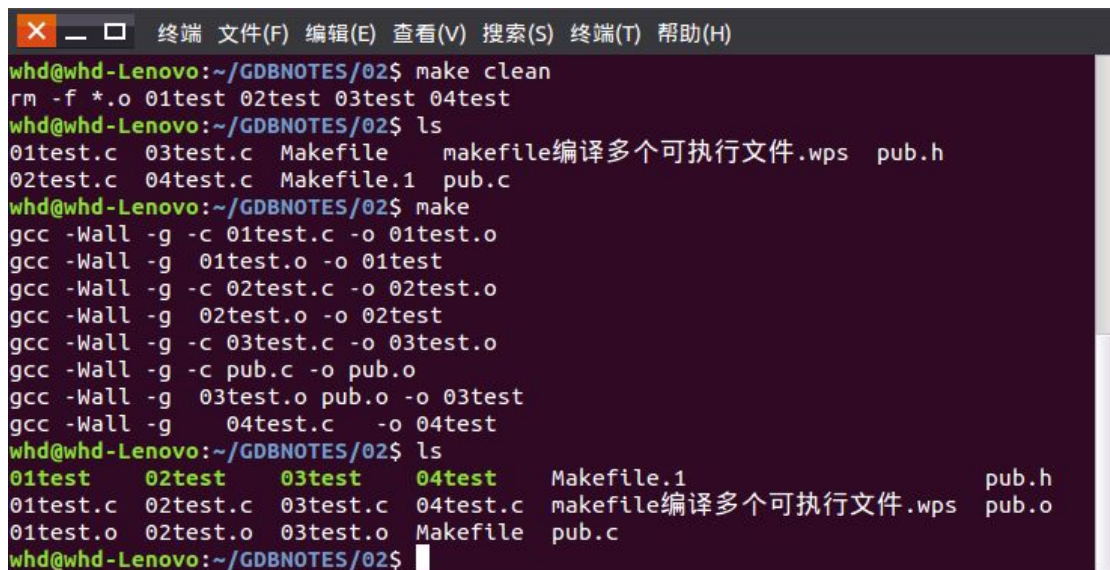
```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c  03test.c  Makefile.1                pub.c
02test.c  Makefile  makefile编译多个可执行文件.wps  pub.h
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
gcc -Wall -g -c 03test.c -o 03test.o
gcc -Wall -g -c pub.c -o pub.o
gcc -Wall -g 03test.o pub.o -o 03test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test    02test    03test    Makefile                pub.c
01test.c  02test.c  03test.c  Makefile.1              pub.h
01test.o  02test.o  03test.o  makefile编译多个可执行文件.wps  pub.o
whd@whd-Lenovo:~/GDBNOTES/02$
```

在比如我我现在要编写一个程序 04test,修改 Makefile;



```
.PHONY:clean all
CC=gcc
CFLAGS=-Wall -g
BIN=01test 02test 03test 04test
all:$(BIN)
#%.o:%.c
# $(CC) $(CFLAGS) -c $^ -o $@
%.o:
$(CC) $(CFLAGS) -c $^ -o $@
01test:01test.o
$(CC) $(CFLAGS) $^ -o 01test
02test:02test.o
$(CC) $(CFLAGS) $^ -o 02test
03test:03test.o pub.o
$(CC) $(CFLAGS) $^ -o 03test
clean:
rm -f *.o $(BIN)
```

运行结果为:



```
whd@whd-Lenovo:~/GDBNOTES/02$ make clean
rm -f *.o 01test 02test 03test 04test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test.c 03test.c Makefile makefile编译多个可执行文件.wps pub.h
02test.c 04test.c Makefile.1 pub.c
whd@whd-Lenovo:~/GDBNOTES/02$ make
gcc -Wall -g -c 01test.c -o 01test.o
gcc -Wall -g 01test.o -o 01test
gcc -Wall -g -c 02test.c -o 02test.o
gcc -Wall -g 02test.o -o 02test
gcc -Wall -g -c 03test.c -o 03test.o
gcc -Wall -g -c pub.c -o pub.o
gcc -Wall -g 03test.o pub.o -o 03test
gcc -Wall -g 04test.c -o 04test
whd@whd-Lenovo:~/GDBNOTES/02$ ls
01test 02test 03test 04test Makefile.1 pub.h
01test.c 02test.c 03test.c 04test.c makefile编译多个可执行文件.wps pub.o
01test.o 02test.o 03test.o Makefile pub.c
whd@whd-Lenovo:~/GDBNOTES/02$
```

我们在学习 c 语言或者 C++语言的时候,使用 Makefile 比较方便。