<https://blog.csdn.net/xukai871105/article/details/36476793>

0.前言

    从学习C语言开始就慢慢开始接触makefile，查阅了很多的makefile的资料但总感觉没有真正掌握makefile，如果自己动手写一个makefile总觉得非常吃力。所以特意借助博客总结makefile的相关知识，通过例子说明makefile的具体用法。

    例说makefile分为以下几个部分，更多内容请参考【[例说makefile索引博文](http://blog.csdn.net/xukai871105/article/details/37083675)】

    1.只有单个C文件

    2.含有多个C文件

    3.需要包括头文件路径

    4.增加宏定义

    5.增加系统共享库

    6.增加自定义共享库

    7.一个实际的例子

    【代码仓库】——[makefile-example](https://bitbucket.org/xukai871105/makefile-example" \t "_blank)

    代码仓库位于bitbucket，可借助TortoiseHg（GUI工具）克隆代码或者在网页中直接下载zip包。

1.三个C文件和三个头文件

    此处的例子稍微复杂些但更接近实际情况。

    文件结果如下：根目录中包含test.c makefileh和文件夹test-add和文件夹test-sub。

test.c makefile

    【test-add】test-add.c test-add.h

    【test-sub】test-sub.c test-sub.h

    【test.c】

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. #include <stdio.h>
2. #include <test-add.h>
3. #include <test-sub.h>
4. **int** main(**void**)
5. {
6. **int** a = 3;
7. **int** b = 2;
9. printf("a=%d\n", a);
10. printf("b=%d\n", b);
12. printf("a+b=%d\n", add(a,b));
13. printf("a-b=%d\n", sub(a,b));
14. **return** 0;
15. }

    【test-add.c】

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. #include <test-add.h>
2. **int** add(**int** a, **int** b)
3. {
4. **return** a+b;
5. }

    【test-add.h】

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. #ifndef \_\_TEST\_ADD
2. **int** add(**int** a, **int** b);
3. #endif

    【test-sub.c】

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. #include "test-sub.h"
2. **int** sub(**int** a, **int** b)
3. {
4. **return** a-b;
5. }

    【test-sub.h】

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. #ifndef \_\_TEST\_SUB
2. **int** sub(**int** a, **int** b);
3. #endif

2.复习gcc指令

    gcc指令可通过-I前缀指定头文件路径，特别说明./代表当前路径，../代表上一级目录。

3.编写makefile

    请替换其中的[tab]，并以代码仓库中的makefile文件为主。

**[cpp]** [view plain](https://blog.csdn.net/xukai871105/article/details/36476793) [copy](https://blog.csdn.net/xukai871105/article/details/36476793)

1. # 指令编译器和选项
2. CC=gcc
3. CFLAGS=-Wall -std=gnu99
5. # 目标文件
6. TARGET=test
7. SRCS = test.c \
8. ./test-add/test-add.c \
9. ./test-sub/test-sub.c
11. INC = -I./test-add -I./test-sub
13. OBJS = $(SRCS:.c=.o)
15. $(TARGET):$(OBJS)
16. #   @echo TARGET:$@
17. #   @echo OBJECTS:$^
18. [tab]$(CC) -o $@ $^
20. clean:
21. [tab]rm -rf $(TARGET) $(OBJS)
23. %.o:%.c
24. [tab]$(CC) $(CFLAGS) $(INC) -o $@ -c $<

    【具体说明】

    【1】相比于单个文件和多个文件的makefile，通过变量INC制定了头文件路径。头文件路径之间通过空格隔开。

    【2】编译规则%.o:%.c中加入了头文件参数$(CC) $(CFLAGS) $(INC) -o $@ -c $<，那么在编译的过程中便会出现

    gcc -Wall -std=gnu99 -I./test-add -I./test-sub -o test.o -c test.c。和单个文件和多个文件的makefile相比增加了头文件路径参数。

    【3】SRCS变量中，文件较多时可通过“\”符号续行。

    【编译】

    make clean && make

    【控制台输出】

rm -rf test test.o ./test-add/test-add.o ./test-sub/test-sub.o

gcc -Wall -std=gnu99 -I./test-add -I./test-sub -o test.o -c test.c

gcc -Wall -std=gnu99 -I./test-add -I./test-sub -o test-add/test-add.o -c test-add/test-add.c

gcc -Wall -std=gnu99 -I./test-add -I./test-sub -o test-sub/test-sub.o -c test-sub/test-sub.c

gcc -o test test.o test-add/test-add.o test-sub/test-sub.o

    从控制台的输出可以看出，通过make clean清除上一次的可执行文件和目标文件，然后依次编译各个C文件，在编译的过程中制定了头文件路径，最后把3个目标文件链接为最终可执行文件。