# make多文件编译工具

实验名称：make多文件编译工具

学时安排：2课时 指导老师：李赞

实验类别：验证型、设计型 实验要求：1人1组

学号： 姓名： 班级：

## 一、实验目的和任务

1. 掌握make工具的基础命令和应用。

2. 使用make工具编译有多文件组成的小型工程。

## 二、实验设备介绍

1.软件需求： win10操作系统，VMware workstation，ubuntu18。

## 三、注意事项和要求

1.保持源文件和make文件放置在正确的文件目录中。

2.掌握makefile文件编写规则。

## 四、实验内容和步骤

### 4.1 Linux中的Make编译与makefile

对于只有单个文件的hello.c源程序，一个编译命令就可以将其生成为可执行文件。对于一个规模较大的项目，不仅要逐个编译每个源文件，而且文件之间具有较复杂的条件限制和额外操作，编译过程还要执行shell脚本，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，程中的哪些源文件需要编译以及如何编译、需要创建那些库文件以及如何创建这些库文件、如何最后产生我们想要的可执行文件。复杂的依赖关系和编译命令与shell脚本操作综合在一起用一个文件记录下来命名为makefile文件[该文件没有后缀名，内容是普通的文本文件]。用户调用make命令时，make工具查找并读取makefile文件中的内容，根据文件内容按照规则进行依赖检查，执行指令，最后生成目标文件。makefile可看成大项软件项目编译过程的批处理。

一个 Makefile 中通常包含下面内容：

1.需要由 make 工具创建的目标体（target），通常是目标文件或可执行文件。

2.要创建的目标体所依赖的文件（dependency\_file）。

3.创建每个目标体时需要运行的命令（command）。

makefile文件中的命令格式如下：

target：dependency\_files

[TAB]command

注意：每一个命令行必须以[Tab]字符开始，[Tab]字符告诉 make 此行是一个命令行。Makefile 中的目标"clean"没有依赖，只有命令。它所指定的命令用来删除 make 过程产生的中间文件（清理工作）。makefile撰写时一般最终的生成文件放置在文件最开始，而其依赖的文件还没有生成，因此makefile会先将依赖关系入栈，直到满足条件的依赖关系存在就执行相应的编译指令，然后再从栈中执行相应的编译命令，最首的目标文件一般是最后才最终执行编译。

### 4.2make编译多文件工程示例一

现有一个多文件的项目，项目根目录有main.c和makefile文件，add目录和sub目录，add目录中有add\_int.c和add\_float.c文件，sub目录有sub\_int.c和sub\_float.c文件，参看下面的项目结构图。

//《Linux网络编程》page33

Makefile文件的内容：

#生成cacu, [目标:依赖] 目标文件，可以是Object File，也可以是执行文件。还可以是一个标签（Label）

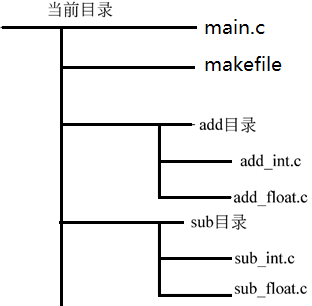
cacu:add\_int.o add\_float.o sub\_int.o sub\_float.o main.o

#编译命令 gcc

gcc -o cacu add/add\_int.o add/add\_float.o sub/sub\_int.o sub/sub\_float.o main.o

#生成add\_int.o的规则，[目标:依赖]

add\_int.o:add/add\_int.c add/add.h

#编译命令 gcc 生成add\_int.o

gcc -c -o add/add\_int.o add/add\_int.c

#生成add\_float.o的规则，[目标:依赖]

add\_float.o:add/add\_float.c add/add.h

#编译命令 gcc 生成add\_int.o

gcc -c -o add/add\_float.o add/add\_float.c

#生成sub\_int.o的规则，[目标:依赖]

sub\_int.o:sub/sub\_int.c sub/sub.h

#编译命令 gcc 生成sub\_int.o

gcc -c -o sub/sub\_int.o sub/sub\_int.c

#生成sub\_float.o的规则，[目标:依赖]

sub\_float.o:sub/sub\_float.c sub/sub.h

#编译命令 gcc 生成sub\_float.o

gcc -c -o sub/sub\_float.o sub/sub\_float.c

#生成main.o的规则，[目标:依赖]

sub\_float.o:main.c add/add.h sub/sub.h

#编译命令 gcc 生成main.o -I参数是用来指定头文件目录

gcc -c -o main.o main.c -I add -I sub

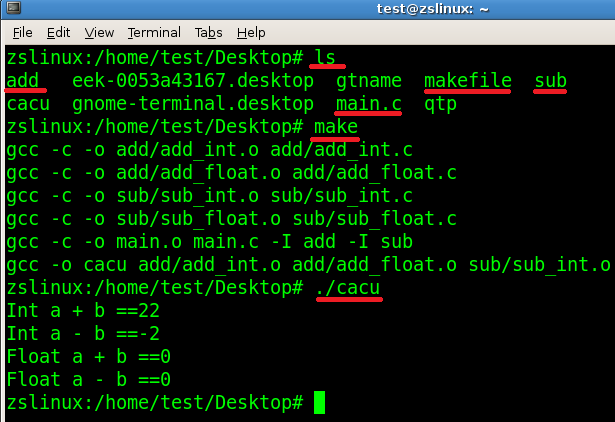
#清理规则 执行make clean命令会调用该规则

#删除这些文件包括可执行文件 cacu 中间文件 .o文件

clean:

rm -f cacu add/add\_int.o add/add\_float.o sub/sub\_int.o sub/sub\_float.o main.o

作业要求：准备相应的项目文件，并编辑makefile文件内容，运行make命令时，将逐行执行相应gcc命令，最后生成cacu文件，注意cacu是在最后才执行的。



### 4.3make编译多文件工程示例二

当前目录下有一个工程共有五个源文件mytool1.c、mytool1.h；mytool2.c、mytool2.h；main.c；main.c 引用的头文件 #include "mytool1.h" #include "mytool2.h"

mytool1.c引用的头文件 #include "mytool1.h"

mytool2.c引用的头文件 #include "mytool2.h"

常规法写第一个 Makefile

main:main.o mytool1.o mytool2.o

[tab]gcc -o main main.o mytool1.o mytool2.o

main.o:main.c mytool1.h mytool2.h

[tab]gcc -c main.c

mytool1.o:mytool1.c mytool1.h

[tab]gcc -c mytool1.c

mytool2.o:mytool2.c mytool2.h

[tab]gcc -c mytool2.c

clean:

[tab]rm -f \*.o main

使用变量：

OFILES=main.o mytool1.o mytool2.o

main:$(OFILES)

gcc -o main $(OFILES)

main.o:main.c mytool1.h mytool2.h

gcc -c main.c

mytool1.o:mytool1.c mytool1.h

gcc -c mytool1.c

mytool2.o:mytool2.c mytool2.h

gcc -c mytool2.c

clean:

rm -f main $(OFILES)

使用自动推导：

让make工具自动在.o和.c文件之间建立关联，自动为.o文件添加.c文件依赖。所以在makefile中就不再出现同名的.c文件，还有更重要的是gcc命令也不需要写，make工具会自动添加gcc命令。通常变量都使用大写字母。

CC=gcc

OFILES=main.o mytool1.o mytool2.o

main:$(OFILES)

$(CC) -o main $(OFILES)

main.o:mytool1.h mytool2.h

mytool1.o: mytool1.h

mytool2.o: mytool2.h

clean:

rm -f main $(OFILES)

### 4.4 make常见问题

#### 4.4.1make: Nothing to be done for `all' 解决方法

1.这句提示是说明程序在运行本次make前已经编译好了，源代码没有进行任何改动，不需要调用gcc等工具对源码进行编译。希望通过重新编译，查看过程中的信息，可以先删除以前编译产生的目标文件：

make clean

然后

make

## 五、程序调试中遇到的问题和解决过程及运行结果

1.按照步骤完成多文件编译过程，并将结果截图保存。

2.描述编译过程中出现的问题及解决过程。