



Programming  
Your Future

# 第五章 Makefile写法及自动化工具

东软IT人才实训中心

Programming Your Future

# 第五章：Makefile写法及自动化工具

## 目标：

本章使学员熟练掌握linux操作系统下makefile的写法，通过本课的学习，学员应该掌握如下知识：

- ☑ 了解makefile的语法及规则
- ☑ 掌握利用makefile文件进行编译
- ☑ 了解自动makefile文件生成原理

学时：3 学时

教学方法：讲授ppt+上机操作+实例演示

## 5.1 make工具介绍

- make是一个项目维护工具，能够保证软件由最新的模块构成。为了达到软件的及时编译，链接更新。
- make需要得到两方面的信息：
  1. 关于可执行文件和各程序模块间的相互关系。
  2. 文件的修改日期。

## 5.2 Makefile依赖关系描述

- 一般的语法规则为：

target:dependency

command

- 例如，对于之前的例程1，其 Makefile 可书写如下：

```
1 #Makefile for exp1
2 hello: hello.c
3         gcc -o hello hello.c
```

- 目标文件为hello，其依赖于hello.c，其生成指令为 gcc -o hello hello.c。在终端下执行 make，会自动生成可执行文件 hello。

```
root@neusoft-vm:~/exp1# make
gcc -o hello hello.c
root@neusoft-vm:~/exp1# ls
hello hello.c Makefile
root@neusoft-vm:~/exp1# ./hello
Hello World!
```

## 5.3 Makefile实例

```
//main.c
#include "main.h"
int main()
{
    add();
    del();
    modify();
    return 0;
}
```

```
//main.h
void add();
void del();
void modify();
```

```
//add.c
#include <stdio.h>
void add()
{
    printf("add\n");
}
```

```
//del.c
#include <stdio.h>
void del()
{
    printf("del\n");
}
```

```
//modify.c
#include <stdio.h>
void modify()
{
    printf("modify\n");
}
```

对于之前的例程，其Makefile可书写如下：

```
1 #Makefile for exp2
2 main: main.o add.o del.o modify.o
3     gcc -o main main.o add.o del.o modify.o
4 main.o: main.c
5     gcc -c main.c
6 add.o: add.c
7     gcc -c add.c
8 del.o: del.c
9     gcc -c del.c
10 modify.o: modify.c
11     gcc -c modify.c
```

## 5.4 Makefile变量替换

使用变量替换，可以增加 Makefile 的可读性，降低编写 Makefile 时出错的机率。

```
example: example.c  
(TAB键) cc -o example example.c
```

可使用变量替换如下：

```
OBJ = example  
DEPENDENCIES = example.c  
CCFLAGS= -o  
$(OBJ):$(DEPENDENCIES)  
(TAB键) gcc $(CCFLAGS) $(OBJ) $(DEPENDENCIES)
```



## 5.5 Makefile自动化变量

- 一些常见的自动化变量说明如下：
  - (1) **\$@** — 目标文件的名称；
  - (2) **\$\$** — 所有的依赖文件，以空格分开，不包含重复的依赖文件；
  - (3) **\$<** — 第一个依赖文件的名称；
  - (4) **CC** — C编译器的名称，默认值为cc；
  - (5) **CCFLAGS** — C编译器的选项。

## 5.6 自动化变量的使用

```
1 #Makefile for exp2
2 main: main.o add.o del.o modify.o
3     gcc -o main main.o add.o del.o modify.o
4 main.o: main.c
5     gcc -c main.c
6 add.o: add.c
7     gcc -c add.c
8 del.o: del.c
9     gcc -c del.c
10 modify.o: modify.c
11     gcc -c modify.c
```

```
1 #Makefile2 for exp2
2 main: main.o add.o del.o modify.o
3     gcc -o $@ $^
4 main.o: main.c
5     gcc -c $<
6 add.o: add.c
7     gcc -c $<
8 del.o: del.c
9     gcc -c $<
10 modify.o: modify.c
11     gcc -c $<
```

```
1 #Makefile3 for exp2
2 main: main.o add.o del.o modify.o
3     gcc -o $@ $^
4 .c.o:
5     gcc -c $<
```



## 5.7 makefile目标编译

- 如果不指定目标 (target) make会默认第一个target

- 规范的makefile文件都有以下常见的几个目标：

`make all` - 编译所有目标。

`make clean` - 在编译结束后删除.o文件。


`make install` - 编译结束后将最终的可执行文件安装到系统的某一个位置。

```
#makefile for example
example: example.o add.o modify.o delete.o
        $(CC) -o $e $^
.c.o:
        $(CC) -c $<
all: example
clean: all
        rm -f *.o
install: clean
        cp example /usr/local/bin
```

## 5.8 makefile自动化工具

### GNU Autotools

用于自动生成makefile文件主要有二个工具：

**Autoconf** - 这个工具用来生成configure脚本。这个脚本主要用来分析你的系统以找到合适的工具使用。 

**Automake** - 这个工具用来生成Makefile相关文件。它需要用到Autoconf提供的信息。譬如，如果Autoconf检测到你的系统使用“gcc”，那Makefile就使用gcc作为C编译器。反之，如果找到“cc”，那就使用“cc”。



## 5.8 makefile自动化工具

- CMake

跨平台的安装(编译)工具, 能够输出各种各样的makefile或者project文件, 能测试编译器所支持的C++特性, 类似于automake。

Cmake不再使你在构建项目时郁闷地想自杀了。

一位KDE开发者



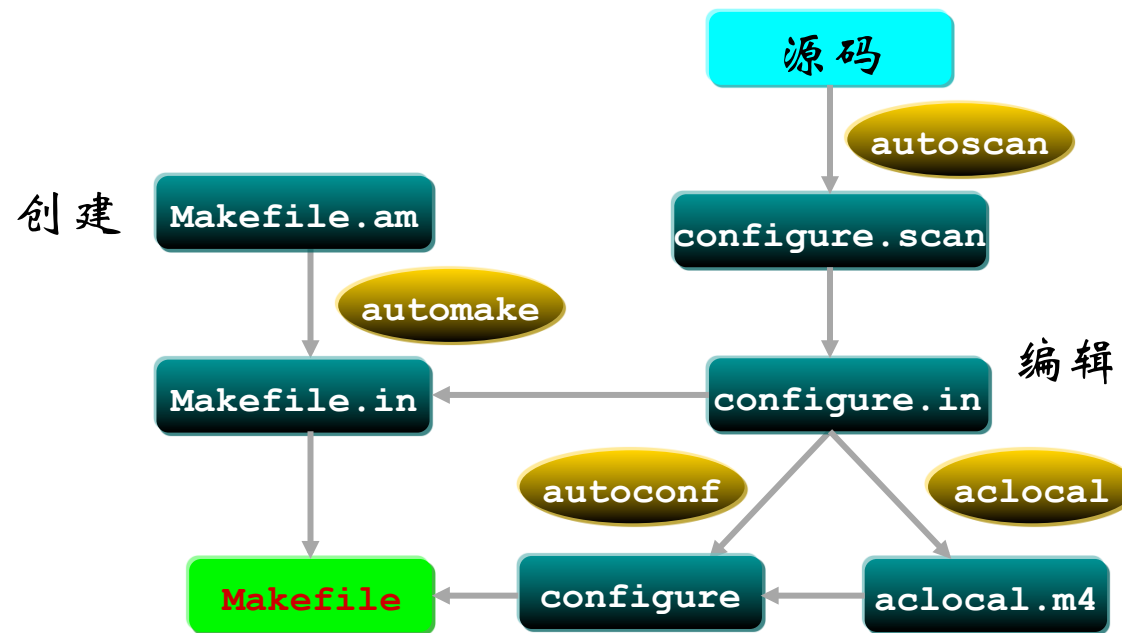
## 5.8.1 GNU Autotools应用

GNU Autotools指的就是下面的六个工具：

1. aclocal
2. autoscan
3. autoconf
4. autoheader
5. automake
6. libtool

## 5.8.1 GNU Autotools应用

- 使用autotools工具创建makefile参考流程图



## 5.8.1 GNU Autotools应用

Autotools的使用流程：

- 手工编写`Makefile.am`这个文件。
- 在源代码目录树的最高层运行`autoscan`。然后手动修改`configure.scan`文件，并改名为`configure.in`。
- 运行`aclocal`，它会根据`configure.in`的内容生成`aclocal.m4`文件。
- 运行`autoconf`，它根据`configure.in`和`aclocal.m4`的内容生成`configure`这个配置脚本文件。
- 运行`automake --add-missing`，它根据`Makefile.am`的内容生成`Makefile.in`。
- 运行`configure`，它会根据`Makefile.in`的内容生成`Makefile`这个文件。

获得`Makefile`文件后，就可以使用`make`程序来管理我们的工程。



### 5.8.1 GNU Autotools应用

- Autotools使用实例:

建立makefile.am文件。

```
AUTOMAKE_OPTIONS=foreign
bin_PROGRAMS=main
main_SOURCES=main.c add.c del.c modify.c
```

~~~~~

```
"Makefile.am" 3L, 84C
```

1,1

全部

## 5.8.1 GNU Autotools应用

运行autoscan。然后手动修改configure.scan文件，并改名为configure.in。

```
neusoft@neusoft-desktop:~/mypro/maketest$ autoscan
neusoft@neusoft-desktop:~/mypro/maketest$ ls
add.c          configure.scan  main.c  makefile1  makefile3  modify.c
autoscan.log   del.c          main.h  makefile2  Makefile.am
```

```
#                                     -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.63])
AC_INIT([FULL-PACKAGE-NAME], [VERSION], [BUG-REPORT-ADDRESS])
AC_CONFIG_SRCDIR([main.c])
AC_CONFIG_HEADERS([config.h])

# Checks for programs.
AC_PROG_CC

# Checks for libraries.

# Checks for header files.

# Checks for typedefs, structures, and compiler characteristics.

# Checks for library functions.

AC_CONFIG_FILES([Makefile])
AC_OUTPUT
~
~
"configure.scan" 21L, 494C          1,1          全部
```

## 5.8.1 GNU Autotools应用

```
##                                     -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.67])
AC_INIT([FULL-PACKAGE-NAME], [VERSION], [BUG-REPORT-ADDRESS])
AC_CONFIG_SRCDIR([main.c])
#AC_CONFIG_HEADERS([config.h])
AM_INIT_AUTOMAKE(main,1,0)

# Checks for programs.
AC_PROG_CC

# Checks for libraries.

# Checks for header files.

# Checks for typedefs, structures, and compiler characteristics.

# Checks for library functions.

#AC_CONFIG_FILES([makefile])
AC_OUTPUT(makefile)
```

执行aclocal 和 autoconf, 分别生成aclocal.m4 和 configure两个文件。

## 5.8.1 GNU Autotools应用

执行 `automake --add-missing`, `automake` 会根据 `makefile.am` 产生一些文件, 包含了最重要的 `makefile.in`。然后运行 `configure`。

```
neusoft@neusoft-desktop:~/mypro/maketest$ ls
aclocal.m4      autoscan.log  del.c          main.c         makefile2      Makefile.in
add.c           configure     depcomp        main.h         makefile3      missing
autom4te.cache  configure.in  install-sh     makefile1      Makefile.am    modify.c
neusoft@neusoft-desktop:~/mypro/maketest$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
configure: creating ./config.status
config.status: creating Makefile
config.status: executing depfiles commands
neusoft@neusoft-desktop:~/mypro/maketest$
```

## 5.8.1 GNU Autotools应用

```
neusoft@neusoft-desktop:~/mypro/maketest$ make
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -D
PACKAGE_VERSION=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME VERSION\" -DPACKA
GE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -I. -g -O2 -MT main.o -MD -MP -MF .deps/mai
n.Tpo -c -o main.o main.c
mv -f .deps/main.Tpo .deps/main.Po
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -D
PACKAGE_VERSION=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME VERSION\" -DPACKA
GE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -I. -g -O2 -MT add.o -MD -MP -MF .deps/add.
Tpo -c -o add.o add.c
mv -f .deps/add.Tpo .deps/add.Po
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -D
PACKAGE_VERSION=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME VERSION\" -DPACKA
GE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -I. -g -O2 -MT del.o -MD -MP -MF .deps/del.
Tpo -c -o del.o del.c
mv -f .deps/del.Tpo .deps/del.Po
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -D
PACKAGE_VERSION=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME VERSION\" -DPACKA
GE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -I. -g -O2 -MT modify.o -MD -MP -MF .deps/m
odify.Tpo -c -o modify.o modify.c
mv -f .deps/modify.Tpo .deps/modify.Po
gcc -g -O2 -o main main.o add.o del.o modify.o
neusoft@neusoft-desktop:~/mypro/maketest$
```

## 5.8.2 cmake

- CMake使用方法

CMake的所有语句都写在`CMakeLists.txt`的文件中。

当`CMakeLists.txt`文件确定后,可以用`ccmake`命令对相关 的变量值进行配置。这个命令必须指向`CMakeLists.txt`所在的目录。

配置完成之后,应用`cmake`命令生成相应的`makefile`。

- 命令执行

```
#ccmake
```

```
#cmake
```



## 5.8.2 cmake

- cmake使用例程1:

对于只有一个源码文件的工程如何通过cmake的帮助构建makefile。

- 工程目录中的文件结构

```
neusoft@neusoft-desktop:~/mypro/cmaketest/test1$ tree
.
|-- CMakeLists.txt
|-- build
|-- inc
|   |-- hello.h
|-- obj
|-- src
|   |-- hello.c
4 directories, 3 files
```

## 5.8.2 cmake

- 在工程目录中创建CMakeLists.txt

```
1 #project name
2 PROJECT(hello)
3 #head file path
4 INCLUDE_DIRECTORIES(inc)
5 #source_directory
6 AUX_SOURCE_DIRECTORY(src DIR_SRCS)
7 #set environment variable
8 SET(HELLO ${DIR_SRCS})
9 #add executable file
10 ADD_EXECUTABLE(..../obj/hello ${DIR_SRCS})
```

第2行 工程名称为hello

第4行 头文件所在路径为当前路径下的inc

第6行 源码文件所在路径为当前路径下的src

第8行 设定环境HELLO变量为源码文件所在路径

第10行 生成执行文件所存放路径及文件名

## 5.8.2 cmake

- cmake使用例程2:

利用之前例中的myalib.c 制作成动态库文件libtest.so, 结合动态库编译的工程如何配置cmake环境。

- 工程目录中的文件结构

```
neusoft@neusoft-desktop:~/mypro/cmaketest/test2$ tree
.
|-- CMakeLists.txt
|-- build
|-- inc
|   |-- myalib.h
|-- lib
|   |-- libtest.so
|-- obj
|-- src
|   |-- main.c
5 directories, 4 files
```

## 5.8.2 cmake

- CMakeLists.txt 配置

```
1 #project name
2 PROJECT(main)
3 #head file path
4 INCLUDE_DIRECTORIES(inc)
5 #lib file path
6 LINK_DIRECTORIES(lib)
7 #source directory
8 AUX_SOURCE_DIRECTORY(src DIR_SRCS)
9 #set environment variable
10 SET(LIBRARIES libtest.so)
11 #add executable file
12 ADD_EXECUTABLE(..../obj/main ${DIR_SRCS})
13 #add lib file
14 TARGET_LINK_LIBRARIES(..../obj/main ${LIBRARIES})
```

第6行 链接库文件libtest.so所在路径

第10行 声明链接库变量的值为libtest.so

第14行 链接libtest.so生成main执行文件

## 5.8.2 cmake

- cmake运行过程

为支持外部编译，创建build目录，在此目录里进行makefile配置过程。

```
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/build$ cmake ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
CMake Warning (dev) in CMakeLists.txt:
  No cmake_minimum_required command is present.  A line of code such as

    cmake_minimum_required(VERSION 2.6)

  should be added at the top of the file.  The version specified may be lower
  if you wish to support older CMake versions for this project.  For more
  information run "cmake --help-policy CMP0000".
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done
-- Generating done
-- Build files have been written to: /home/neusoft/mypro/cmaketest/test2/build
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/build$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile  obj
```

- 外部编译的好处是不影响工程目录里文件结构。

## 5.8.2 cmake

- 测试makefile使用

```
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/build$ make
Scanning dependencies of target main
[100%] Building C object CMakeFiles/./obj/main.dir/src/main.o
Linking C executable ../obj/main
[100%] Built target ../obj/main
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/build$ ls
CMakeCache.txt CMakeFiles cmake_install.cmake Makefile obj
```

```
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/build$ cd ../obj/
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/obj$ ls
main
neusoft@neusoft-desktop:~/mypro/cmaketest/test2/obj$ ./main
Hi, test!
```



# 术语

| 缩语、术语 | 英文全称 | 解释 |
|-------|------|----|
|       |      |    |
|       |      |    |
|       |      |    |
|       |      |    |
|       |      |    |
|       |      |    |

# Neusoft

Beyond Technology

Programming Your Future