

# 一、简介

作为Linux下的程序开发人员，一定都遇到过Makefile，用make命令来编译自己写的程序确实是很方便。一般情况下，大家都是手工写一个简单Makefile，如果要想写出一个符合自由软件惯例的Makefile就不那么容易了。在本文中，将介绍如何使用autoconf和automake两个工具来帮助我们自动地生成符合自由软件惯例的Makefile，这样就可以象常见的GNU程序一样，只要使用“./configure”，“make”，“make instal”就可以把程序安装到Linux系统中去了。这将特别适合想做开放源代码软件的程序开发人员，又或如果你只是自己写些小的Toy程序，那么这篇文章对你也会有很大的帮助。

## 一、Makefile介绍

Makefile是用于自动编译和链接的，一个工程有很多文件组成，每一个文件的改变都会导致工程的重新链接，但是不是所有的文件都需要重新编译，Makefile中纪录有文件的信息，在make时会决定在链接的时候需要重新编译哪些文件。

Makefile的宗旨就是：让编译器知道要编译一个文件需要依赖其他的哪些文件。当那些依赖文件有了改变，编译器会自动的发现最终的生成文件已经过时，而重新编译相应的模块。

Makefile的基本结构不是很复杂，但当一个程序开发人员开始写Makefile时，经常会怀疑自己写的是否符合惯例，而且自己写的Makefile经常和自己的开发环境相关联，当系统环境变量或路径发生了变化后，Makefile可能还要跟着修改。这样就造成了手工书写Makefile的诸多问题，automake恰好能很好地帮助我们解决这些问题。

使用automake，程序开发人员只需要写一些简单的含有预定义宏的文件，由autoconf根据一个宏文件生成configure，由automake根据另一个宏文件生成Makefile.in，再使用configure依据Makefile.in来生成一个符合惯例的Makefile。下面我们将详细介绍Makefile的automake生成方法。

# 二、实验

## 1、建目录

在你的工作目录下建一个helloworld目录，我们用它来存放helloworld程序及相关文件：

```
$ mkdir helloworld
```

```
$ cd helloworld
```

```
$ touch hello.c
```

## 2、编写hello.c

```
int main(int argc, char** argv)
{
    printf("Hello, Linux World! ");
    return 0;
}
```

完成后保存退出。现在在helloworld目录下就应该有一个你自己写的hello.c了。

```
ly@ly-VirtualBox:~/helloworld$ ls
hello.c
ly@ly-VirtualBox:~/helloworld$
```



### 3.使用autoscan工具生成configure.scan文件

```
ly@ly-VirtualBox:~/helloworld$ autoscan ./
ly@ly-VirtualBox:~/helloworld$ ls
autoscan.log  configure.scan  hello.c
ly@ly-VirtualBox:~/helloworld$ cat configure.scan
#                                     -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.68])
AC_INIT([FULL-PACKAGE-NAME], [VERSION], [BUG-REPORT-ADDRESS])
AC_CONFIG_SRCDIR([hello.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_OUTPUT
```

<http://blog.csdn.net/gulansheng>

该文件的简要说明如下：

- 1、 AC\_PREREQ宏声明本文件要求的autoconf版本，这里是2.68
- 2、 AC\_INIT定义软件的名称和信息。（ DULL-PACKAGE-NAME为软件名， VERSION为软件的版本号， BUG-REPORT-ADDRESS为bug的报告地址，一般为软件作者的邮箱 ）
- 3、 AC\_CONFIG\_SRCDIR用来侦测指定的源码文件是否存在，确定源码目录的有效性。此处为当前目录下hello.c
- 4、 AC\_CONFIG\_HEADER用于生成config.h文件，以便autoheader使用
- 5、 AC\_PROG\_CC用来指定编译器，以便不指定的时候默认为gcc

6、AC\_OUTPUT用来设定config要产生的文件。如果是Makefile,config会把它检查出来的结果带入Makefile.in文件产生合适的Makefile.

#### 4.获得并且修改configure.ac(或configure.in)

```
$cp configure.scan configure.ac
```

并做一下修改

```
1 #                                     -*- Autoconf -*-
2 # Process this file with autoconf to produce a configure script.
3
4 AC_PREREQ([2.68])
5 AC_INIT(hello, 1.0, xxx@163.com)
6 AM_INIT_AUTOMAKE(hello, 1.0)
7 AC_CONFIG_SRCDIR([hello.c])
8 AC_CONFIG_HEADERS([config.h])
9
10 # Checks for programs.
11 AC_PROG_CC
12
13 # Checks for libraries.
14
15 # Checks for header files.
16
17 # Checks for typedefs, structures, and compiler characteristics.
18
19 # Checks for library functions.
20
21 AC_OUTPUT(Makefile)
```

<http://blog.csdn.net/gulansheng>


这里注意重要的一点：AM\_INIT\_AUTOMAKE宏需要自己进行添加，它是automake所必备的宏。

#### 5.使用aclocal工具生成aclocal.m4

```
ly@ly-VirtualBox:~/helloworld$ aclocal
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4  autom4te.cache  autoscan.log  configure.ac  configure.scan  hello.c
ly@ly-VirtualBox:~/helloworld$
```

<http://blog.csdn.net/gulansheng>

#### 6.使用autoconf工具生成configure文件



```
ly@ly-VirtualBox:~/helloworld$ autoconf
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4      autoscan.log  configure.ac  hello.c
autom4te.cache  configure     configure.scan
ly@ly-VirtualBox:~/helloworld$
```

<http://blog.csdn.net/gulansheng>

#### 7.使用autoheader工具生成config.h.in文件

```
ly@ly-VirtualBox:~/helloworld$ autoheader
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4      autoscan.log  configure      configure.scan
autom4te.cache  config.h.in   configure.ac    hello.c
ly@ly-VirtualBox:~/helloworld$
```



## 8.创建Makefile.am文件

Automake工具会根据config.in中的参量把Makefile.am转换成Makefile.in文件。在使用Automake之前，要先手动建立Makefile.am文件。

```
ly@ly-VirtualBox:~/helloworld$ vi Makefile.am
ly@ly-VirtualBox:~/helloworld$ cat Makefile.am
AUTOMAKE_OPTIONS=foreign
bin_PROGRAMS=hello
hello_SOURCES=hello.c
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4      autoscan.log  configure      configure.scan  Makefile.am
autom4te.cache  config.h.in   configure.ac    hello.c
ly@ly-VirtualBox:~/helloworld$
```

几点需要说明：

- 1、AUTOMAKE\_OPTIONS为设置的Automake选项。它有三种等级提供给用户选择：foreign，gnu,gnits,默认等级为gnu.在此使用foreign，它只检测必须的文件。
- 2、bin\_PROGRAMS定义要产生的执行文件名。如果要产生多个可执行文件，则每个文件名用空格隔开。
- 3、hello\_SOURCES定义为hello这个程序所需要的原始文件。如果其石油多个文件组成的，则必须用空格进行隔开。

## 9.使用Automake生成Makefile.in文件

要使用选项 “--add-missing” 可以让Automake自动添加一些必要的脚本文件。如下



```
ly@ly-VirtualBox:~/helloworld$ automake --add-missing
configure.ac:6: installing `./install-sh'
configure.ac:6: installing `./missing'
Makefile.am: installing `./depcomp'
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4      config.h.in   configure.scan  install-sh      missing
autom4te.cache  configure      depcomp         Makefile.am
autoscan.log    configure.ac   hello.c         Makefile.in
ly@ly-VirtualBox:~/helloworld$
```

## 10.配置

运行自动配置设置文件configure，把Makefile.in编程最终的Makefile



```
ly@ly-VirtualBox:~/helloworld$ ./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 $(MAKE)... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
configure: creating ./config.status
config.status: creating Makefile
config.status: creating config.h
config.status: executing depfiles commands
ly@ly-VirtualBox:~/helloworld$ ls
aclocal.m4      config.h      config.status  configure.scan  install-sh     Makefile.in
autom4te.cache  config.h.in   configure      depcomp        Makefile       missing
autoscan.log    config.log    configure.ac   hello.c        Makefile.am    stamp-h1
ly@ly-VirtualBox:~/helloworld$
```

<http://blog.csdn.net/gulansheng>

## 测试

运行Make命令进行编译。然后运行hello程序

```
ly@ly-VirtualBox:~/helloworld$ make
make all-am
make[1]: Entering directory `/home/ly/helloworld'
gcc -DHAVE_CONFIG_H -I.      -g -O2 -MT hello.o -MD -MP -MF .deps/hello.Tpo -c -o hello.o hello.c
mv -f .deps/hello.Tpo .deps/hello.Po
gcc -g -O2      -o hello hello.o
make[1]: Leaving directory `/home/ly/helloworld'
ly@ly-VirtualBox:~/helloworld$ ./hello
Hello, Linux World! ly@ly-VirtualBox:~/helloworld$
ly@ly-VirtualBox:~/helloworld$
```

<http://blog.csdn.net/gulansheng>



## 三、深入

### 四、深入浅出

针对上面提到的各个命令，我们再做些详细的介绍。

#### 1、autoscan

autoscan是用来扫描源代码目录生成configure.scan文件的。autoscan可以用目录名做为参数，但如果你不使用参数的话，那么autoscan将认为使用的是当前目录。autoscan将扫描你所指定目录中的源文件，并创建configure.scan文件。

## 2、configure.scan

configure.scan包含了系统配置的基本选项，里面都是一些宏定义。我们需要将它改名为configure.in

## 3、aclocal

aclocal是一个perl 脚本程序。aclocal根据configure.in文件的内容，自动生成aclocal.m4文件。aclocal的定义是：“aclocal - create aclocal.m4 by scanning configure.ac”。

## 4、autoconf

autoconf是用来产生configure文件的。configure是一个脚本，它能设置源程序来适应各种不同的操作系统平台，并且根据不同的系统来产生合适的Makefile，从而可以使你的源代码能在不同的操作系统平台上被编译出来。

configure.in文件的内容是一些宏，这些宏经过autoconf 处理后会变成检查系统特性、环境变量、软件必须的参数的shell脚本。configure.in文件中的宏的顺序并没有规定，但是你必须在所有宏的最前面和最后面分别加上AC\_INIT宏和AC\_OUTPUT宏。

在configure.ini中：

#号表示注释，这个宏后面的内容将被忽略。

### AC\_INIT(FILE)

这个宏用来检查源代码所在的路径。

### AM\_INIT\_AUTOMAKE(PACKAGE, VERSION)

这个宏是必须的，它描述了我们将要生成的软件包的名字及其版本号：PACKAGE是软件包的名字，VERSION是版本号。当你使用make dist命令时，它会给你生成一个类似helloworld-1.0.tar.gz的软件发行包，其中就有对应的软件包的名字和版本号。

### AC\_PROG\_CC

这个宏将检查系统所用的C编译器。

### AC\_OUTPUT(FILE)

这个宏是我们输出的Makefile的名字。

我们在使用automake时，实际上还需要用到其他的一些宏，但我们可以用aclocal 来帮我们自动产生。执行aclocal后我们会得到aclocal.m4文件。

产生了configure.in和aclocal.m4 两个宏文件后，我们就可以使用autoconf来产生configure文件了。

## 5、 Makefile.am

Makefile.am是用来生成Makefile.in的，需要你手工书写。Makefile.am中定义了一些内容：

### AUTOMAKE\_OPTIONS

这个是automake的选项。在执行automake时，它会检查目录下是否存在标准GNU软件包中应具备的各种文件，例如AUTHORS、ChangeLog、NEWS等文件。我们将其设置成foreign时，automake会改用一般软件包的标准来检查。

### bin\_PROGRAMS

这个是指定我们所要产生的可执行文件的文件名。如果你要产生多个可执行文件，那么在各个名字间用空格隔开。

### helloworld\_SOURCES

这个是指定产生“helloworld” 时所需要的源代码。如果它用到了多个源文件，那么请使用空格符号将它们隔开。比如需要helloworld.h，helloworld.c那么请写成  
helloworld\_SOURCES= helloworld.h helloworld.c。

如果你在bin\_PROGRAMS定义了多个可执行文件，则对应每个可执行文件都要定义相对的filename\_SOURCES。

## 6、 automake

我们使用automake --add-missing来产生Makefile.in。

选项--add-missing的定义是“add missing standard files to package”，它会让automake加入一个标准的软件包所必须的一些文件。

我们用automake产生出来的Makefile.in文件是符合GNU Makefile惯例的，接下来我们只要执行configure这个shell 脚本就可以产生合适的 Makefile 文件了。

## 7、 Makefile

在符合GNU Makefile惯例的Makefile中，包含了一些基本的预先定义的操作：  
make



根据Makefile编译源代码，连接，生成目标文件，可执行文件。

**make clean**

清除上次的make命令所产生的object文件（后缀为“.o”的文件）及可执行文件。

**make install**

将编译成功的可执行文件安装到系统目录中，一般为/usr/local/bin目录。

**make dist**

产生发布软件包文件（即distribution package）。这个命令将会将可执行文件及相关文件打包成一个tar.gz压缩的文件用来作为发布软件包的软件包。

它会在当前目录下生成一个名字类似“PACKAGE-VERSION.tar.gz”的文件。

PACKAGE和VERSION，是我们在configure.in中定义的

AM\_INIT\_AUTOMAKE(PACKAGE, VERSION)。

**make distcheck**

生成发布软件包并对其进行测试检查，以确定发布包的正确性。这个操作将自动把压缩包文件解开，然后执行configure命令，并且执行make，来确认编译不出现错误，最后提示你软件包已经准备好，可以发布了。

=====  
helloworld-1.0.tar.gz is ready for distribution  
=====

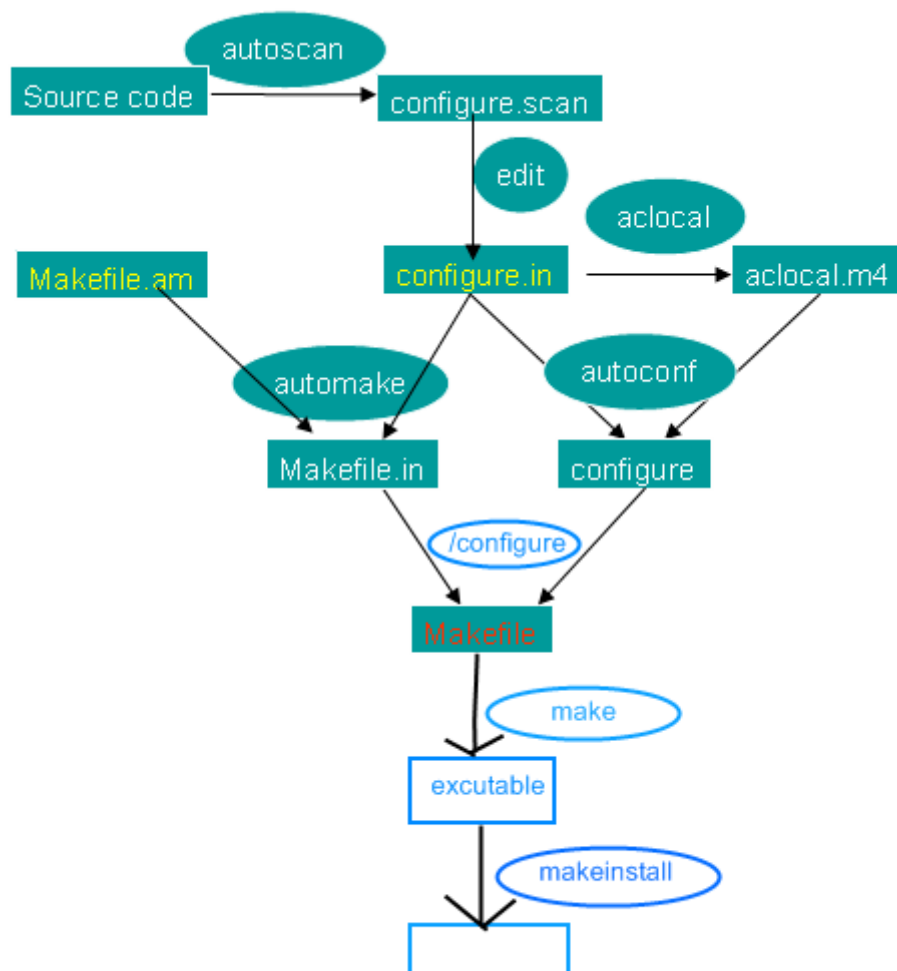
**make distclean**

类似make clean，但同时也将configure生成的文件全部删除掉，包括Makefile。

## 四、过程图示







<http://blog.csdn.net/gulansheng>

## 五、结束语

通过上面的介绍，你应该可以很容易地生成一个你自己的符合GNU惯例的Makefile文件及对应的项目文件。

如果你想写出更复杂的且符合惯例的Makefile，你可以参考一些开放代码的项目中的configure.in和Makefile.am文件，比如：嵌入式数据库sqlite，单元测试cppunit。

```
./configure --host=i386-linux-gnu "CFLAGS=-m32" "CXXFLAGS=-m32"
"LDLFLAGS=-m32 -L/usr/lib/i386-linux-gnu"
```

compile libusb in arm:

```
./configure --host=aarch64-linux-gnu --
prefix=/home/ly/android_tool_source/libusb-1.0.20/out --disable-udev
make
make install
```

compile upgrade\_tool in arm

./configure --host=aarch64-linux-gnu

make