左耳朵耗子写的makefile指南<http://scc.qibebt.cas.cn/docs/linux/base/%B8%FA%CE%D2%D2%BB%C6%F0%D0%B4Makefile-%B3%C2%F0%A9.pdf>

**18 Nov 09 [automake,autoconf使用详解](http://www.laruence.com/2009/11/18/1154.html" \o "Permanent Link to automake,autoconf使用详解)**

* 本文地址: <http://www.laruence.com/2009/11/18/1154.html>
* 文章转自: [http://www.linuxcomputer.cn/](http://www.linuxcomputer.cn/jishuwendang/xinshourumen/200902/03-3029.html)

作为Linux下的程序开发人员,大家一定都遇到过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,将给大家介绍如何使用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,那么这个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.使用的 环境

　　本文所提到的 程序是 基于Linux发行版本：Fedora Core release 1,它包含了我们要用到的 autoconf,automake.

三.从helloworld入手

　　我们从大家最常使用的 例子程序helloworld开始.

　　下面的 过程如果简单地说来就是 ：

　　新建三个文件：

1. helloworld.c
2. configure.in
3. Makefile.am

　　然后执行：

1. aclocal; autoconf; automake --add-missing; ./configure; make; ./helloworld

　　就可以看到Makefile被产生出来,而且可以将helloworld.c编译通过.

　　很简单吧,几条命令就可以做出一个符合惯例的 Makefile,感觉如何呀.

**现在 开始介绍详细的 过程：**

　　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,如在 /home/my/build下：

1. $ mkdir helloword
2. $ cd helloworld

　　2. helloworld.c

　　然后用你自己最喜欢的 编辑器写一个hellowrold.c文件,如命令：vi helloworld.c.使用下面的 代码作为helloworld.c的 内容.

1. #include <stdio.h>
2. int main(int argc, char\*\* argv){
3. printf("%s", 'Hello, Linux World!\n");
4. return 0;
5. }

　　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 helloworld.c了.

　　3.生成configure

　　我们使用autoscan命令来帮助我们根据目录下的 源代码生成一个configure.in的 模板文件.  
　　命令：

1. $ autoscan
2. $ ls
3. configure.scan helloworld.c

　　执行后在 hellowrold目录下会生成一个文件：configure.scan,我们可以拿它作为configure.in的 蓝本.  
　　现在 将configure.scan改名为configure.in,并且编辑它,按下面的 内容修改,去掉无关的 语句：

1. ==========================configure.in内容开始=========================================
2. # -\*- Autoconf -\*-
3. # Process this file with autoconf to produce a configure script.
4. AC\_INIT(helloworld.c)
5. AM\_INIT\_AUTOMAKE(helloworld, 1.0)
6. # Checks for programs.
7. AC\_PROG\_CC
8. # Checks for libraries.
9. # Checks for header files.
10. # Checks for typedefs, structures, and compiler characteristics.
11. # Checks for library functions.
12. AC\_OUTPUT(Makefile)
13. ==========================configure.in内容结束=========================================

　　然后执行命令aclocal和autoconf,分别会产生aclocal.m4及configure两个文件：

1. $ aclocal
2. $ls
3. aclocal.m4 configure.in helloworld.c
4. $ autoconf
5. $ ls
6. aclocal.m4 autom4te.cache configure configure.in helloworld.c

　　大家可以看到configure.in内容是 一些宏定义,这些宏经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.

　　autoconf 是 用来生成自动配置软件源代码脚本（configure）的 工具.configure脚本能独立于autoconf运行,且在 运行的 过程中,不需要用户的 干预.

　　要生成configure文件,你必须告诉autoconf如何找到你所用的 宏.方式是 使用aclocal程序来生成你的 aclocal.m4.

　　aclocal根据configure.in文件的 内容,自动生成aclocal.m4文件.aclocal是 一个perl 脚本程序,它的 定义是 ：”aclocal – create aclocal.m4 by scanning configure.ac”.

　　autoconf从configure.in这个列举编译软件时所需要各种参数的 模板文件中创建configure.

　　autoconf需要GNU m4宏处理器来处理aclocal.m4,生成configure脚本.

　　m4是 一个宏处理器.将输入拷贝到输出,同时将宏展开.宏可以是 内嵌的 ,也可以是 用户定义的 .除了可以展开宏,m4还有一些内建的 函数,用来引用文件,执行命令,整数运算,文本操作,循环等.m4既可以作为编译器的 前端,也可以单独作为一个宏处理器.

4.新建Makefile.am  
　　新建Makefile.am文件,命令：

1. $ vi Makefile.am
2. 内容如下:
3. AUTOMAKE\_OPTIONS=foreign
4. bin\_PROGRAMS=helloworld
5. helloworld\_SOURCES=helloworld.c

　　automake会根据你写的 Makefile.am来自动生成Makefile.in.

　　Makefile.am中定义的 宏和目标,会指导automake生成指定的 代码.例如,宏bin\_PROGRAMS将导致编译和连接的 目标被生成.  
　　5.运行automake:

1. $ automake --add-missing
2. configure.in: installing `./install-sh'
3. configure.in: installing `./mkinstalldirs'
4. configure.in: installing `./missing'
5. Makefile.am: installing `./depcomp'

　　automake会根据Makefile.am文件产生一些文件,包含最重要的 Makefile.in.

　　6.执行configure生成Makefile

1. $./configure
2. checking for a BSD-compatible install... /usr/bin/install -c
3. checking whether build environment is sane... yes
4. checking for gawk... gawk
5. checking whether make sets $(MAKE)... yes
6. checking for gcc... gcc
7. checking for C compiler default output... a.out
8. checking whether the C compiler works... yes
9. checking whether we are cross compiling... no
10. checking for suffix of executables...
11. checking for suffix of object files... o
12. checking whether we are using the GNU C compiler... yes
13. checking whether gcc accepts -g... yes
14. checking for gcc option to accept ANSI C... none needed
15. checking for style of include used by make... GNU
16. checking dependency style of gcc... gcc3
17. configure: creating ./config.status
18. config.status: creating Makefile
19. config.status: executing depfiles commands
20. $ ls -l Makefile
21. -rw-rw-r-- 1 yutao yutao 15035 Oct 15 10:40 Makefile

你可以看到,此时Makefile已经产生出来了.

7.使用Makefile编译代码

1. $make
2. if gcc -DPACKAGE\_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE\_TARNAME=\"full-package-name\" -DPACKAGE\_VERSION=\"VERSION\" -DPACKAGE\_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -DPACKAGE\_BUGREPORT=\"BUG-REPORT-ADDRESS\" -DPACKAGE=\"helloworld\" -DVERSION=\"1.0\" -DSTDC\_HEADERS=1 -DHAVE\_SYS\_TYPES\_H=1 -DHAVE\_SYS\_STAT\_H=1 -DHAVE\_STDLIB\_H=1 -DHAVE\_STRING\_H=1 -DHAVE\_MEMORY\_H=1 -DHAVE\_STRINGS\_H=1 -DHAVE\_INTTYPES\_H=1 -DHAVE\_STDINT\_H=1 -DHAVE\_UNISTD\_H=1 -DHAVE\_STDLIB\_H=1 -I. -I. -g -O2 -MT helloworld.o -MD -MP -MF ".deps/helloworld.Tpo" -c -o helloworld.o helloworld.c; \
3. then mv -f ".deps/helloworld.Tpo" ".deps/helloworld.Po"; else rm -f ".deps/helloworld.Tpo"; exit 1; fi
4. gcc -g -O2 -o helloworld helloworld.o

　　运行helloworld

1. $ ./helloworld
2. Hello, Linux World!

　　这样helloworld就编译出来了,你如果按上面的 步骤来做的 话,应该也会很容易地编译出正确的 helloworld文件.你还可以试着使用一些其他的 make命令,如make clean,make install,make dist,看看它们会给你什么样的 效果.感觉如何？自己也能写出这么专业的 Makefile,老板一定会对你刮目相看.

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 我们要输出的 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autocon

f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 Makefiel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.PA

CKAGE和VERSION,是 我们在 configure.in中定义的 AM\_INIT\_AUTOM

AKE(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.