[Linux下autoconf和automake使用](http://www.cnblogs.com/itech/archive/2010/11/28/1890220.html)

转自：http://hi.baidu.com/liuyanqiong/blog/item/0a6f0ad9d28e1d3d32fa1c7b.html

　　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：　　helloworld.c　　configure.in　　Makefile.am

　　然后执行：autoscan; 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：

      $ mkdir helloword

      $ 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。

　　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：

　　$ autoscan

　　$ ls

　　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。

4，生成configure.in

　  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：

[复制代码](javascript:void(0);)

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
============================configure.in内容开始=========================================  
# -\*- Autoconf -\*-  
  
# Process this file with autoconf to produce a configure script.  
AC\_INIT(helloworld.c)  
AM\_INIT\_AUTOMAKE(helloworld, 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\_OUTPUT(Makefile)  
  
============================configure.in内容结束=========================================

[复制代码](javascript:void(0);)

5 执行aclocal和autoconf

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

[复制代码](javascript:void(0);)

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
$ aclocal  
  
　　$ls  
  
　　aclocal.m4 configure.in helloworld.c  
  
　　$ autoconf  
  
　　$ ls  
  
　　aclocal.m4 autom4te.cache configure configure.in helloworld.c

[复制代码](javascript:void(0);)

　　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.

6、新建Makefile.am

　　新建Makefile.am文件，命令：$ vi Makefile.am

　　内容如下:

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
AUTOMAKE\_OPTIONS=foreign  
  
bin\_PROGRAMS=helloworld  
  
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。

7、运行automake

　　命令：

[复制代码](javascript:void(0);)

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
$ automake --add-missing  
  
configure.in: installing `./install-sh'  
  
configure.in: installing `./mkinstalldirs'  
  
configure.in: installing `./missing'  
  
Makefile.am: installing `./depcomp'

[复制代码](javascript:void(0);)

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

8、执行configure生成Makefile

[复制代码](javascript:void(0);)

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
$ ./configure  
checking for a BSD-compatible installhttp://images.cnblogs.com/dot.gif /usr/bin/install -c  
checking whether build environment is sanehttp://images.cnblogs.com/dot.gif yes  
checking for gawkhttp://images.cnblogs.com/dot.gif gawk  
checking whether make sets $(MAKE)http://images.cnblogs.com/dot.gif yes  
checking for gcchttp://images.cnblogs.com/dot.gif gcc  
checking for C compiler default outputhttp://images.cnblogs.com/dot.gif a.out  
checking whether the C compiler workshttp://images.cnblogs.com/dot.gif yes  
checking whether we are cross compilinghttp://images.cnblogs.com/dot.gif no  
checking for suffix of executableshttp://images.cnblogs.com/dot.gif  
checking for suffix of object fileshttp://images.cnblogs.com/dot.gif o  
checking whether we are using the GNU C compilerhttp://images.cnblogs.com/dot.gif yes  
checking whether gcc accepts -ghttp://images.cnblogs.com/dot.gif yes  
checking for gcc option to accept ANSI Chttp://images.cnblogs.com/dot.gif none needed  
checking for style of include used by makehttp://images.cnblogs.com/dot.gif GNU  
checking dependency style of gcchttp://images.cnblogs.com/dot.gif gcc3  
configure: creating ./config.status  
  
config.status: creating Makefile  
  
config.status: executing depfiles commands  
  
$ ls -l Makefile  
  
-rw-rw-r-- 1 yutao yutao 15035 Oct 15 10:40 Makefile

[复制代码](javascript:void(0);)

你可以看到，此时Makefile已经产生出来了。

9、使用Makefile编译代码

[复制代码](javascript:void(0);)

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
$ make  
  
if gcc -DPACKAGE\_NAME="" -DPACKAGE\_TARNAME="" -DPACKAGE\_VERSION="" -  
  
DPACKAGE\_STRING="" -DPACKAGE\_BUGREPORT="" -DPACKAGE="helloworld" -DVERSION="1.0"  
-I. -I. -g -O2 -MT helloworld.o -MD -MP -MF ".deps/helloworld.Tpo"  
  
-c -o helloworld.o `test -f 'helloworld.c' || echo './'`helloworld.c;  
  
then mv -f ".deps/helloworld.Tpo" ".deps/helloworld.Po";  
  
else rm -f ".deps/helloworld.Tpo"; exit 1;  
  
fi  
  
gcc -g -O2 -o helloworld helloworld.o

[复制代码](javascript:void(0);)

10， 运行helloworld

http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gifCode  
$ ./helloworld  
  
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

　　使用autoconf，根据configure.in和aclocal.m4来产生configure文件。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，根据configure.in和Makefile.am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。

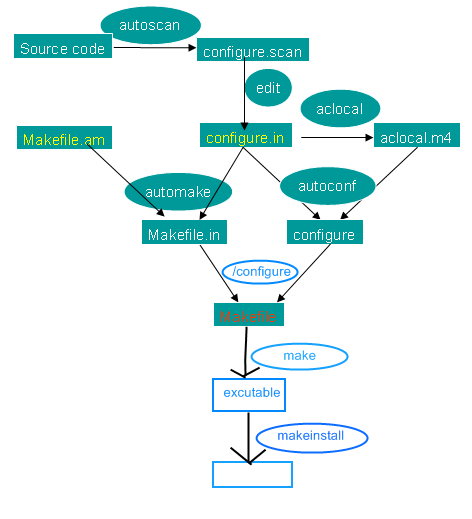
　　make clean　　清除上次的make命令所产生的object文件（后缀为“.o”的文件）及可执行文件。

　　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。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。  
　　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。

五、过程图示



六、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。

七 参考

作者：[iTech](http://itech.cnblogs.com/" \t "_blank)  
出处：[http://itech.cnblogs.com/](http://itech.cnblogs.com/" \t "_blank)   
欢迎转载，欢迎加入qq交流群172758282来讨论build\_release/Linux/Perl/Python/Jenkins！