<https://blog.csdn.net/earbao/article/details/52153600>

转自：<http://blog.chinaunix.net/uid-24704319-id-2594460.html>

本文教你如何使用autoconf、automake等来制作一个以源代码形式(.tar.gz)发布的软件、并可在执行configure时使用自定义参数。

一、概述和基础知识

　　在Linux下得到一个以源代码形式发布的包(一般为.tar.gz或.tar.bz2格式)，我们可以用 ./confiugure、make、make install来编译安装，其中在运行./configure的时候还可以根据自己的需要加入不同的参数(可用./configure --help来查看参数表)。

　　先说说执行./configure后会生成什么东西？运行后系统会根据用户的实际情况生成config.h和多个Makefile。其中 Makefile是运行make时所用的模板；而config.h则会以宏(Marco)的形式记录用户的自定义参数，编译器可以根据config.h来 对源代码进行预编译(pre-compile)，从而生成个性化的执行档。

二、我们的“软件”

　　现在我们可以动手设计一个自己的“软件”了，为了更切合实际，将使用多个源程序，首先建立一个目录tt，用来放我们的东西，然后在tt下建立一个src目录，一般来说源代码都放在src中(好像已经成为一个不成文的规矩了:P)。整体架构如下：　  
　　　|-configure.in  
　　　|-Makefile.am  
　　　|-acconfig.h  
　　　|-src  
　　　　　|-tt.c  
　　　　　|-qq.c  
　　　　　|-qq.h  
　　　　　|-Makefile.am

※说明:  
1. configure.in　这是最重要的文档，整个安装过程都靠它来主导。  
2. Makefile.am　automake会根据它来生成Makefile.in，再由./configure 把Makefile.in变成最终的Makefile，一般来说在顶级目录和各个子目录都应该有一个Makefile.am  
3. acconfig.h　autoheader会根据它来生成config.h.in，再由./configure 把config.h.in变成最终的config.h  
4. tt.c qq.c qq.h　这是我们的源程序。

※源代码内容：

tt.c

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. #include <stdio.h>
2. #include <qq.h>
4. #ifdef HAVE\_CONFIG\_H
5. #include <config.h>
6. #endif
8. **int** main(**void**)
9. {
10. **int** a = 23;
12. printf( "Hello, I am teacher(%d), pls tell me your names!\n", a );
14. #ifdef POPO
15. printf("My name is PoPo!\n");
16. #endif
18. qq();
19. **return** 0;
20. }

qq.c

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. #include <stdio.h>
2. #include <qq.h>
4. #ifdef HAVE\_CONFIG\_H
5. #include <config.h>
6. #endif
8. **void** qq(**void**)
9. {
10. printf("My name is QQ\n");
12. #ifdef POPO
13. printf("QQ: Hey PoPo, long time no see.\n");
14. #endif
15. }

qq.h

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. #ifndef \_\_QQ\_\_
2. #define \_\_QQ\_\_
4. **void** qq(**void**);
6. #endif

※运行流程：  
　1. 首先老师来点名。  
　2. 如果PoPo有来的话，将会报出自己的名字。  
　3. 接著轮到QQ报到，如果PoPo有来的话，QQ会向PoPo问好。

　　显然易见，PoPo是否出席，完全取决于POPO这个宏(Macro)有否被定义，我们只要在编译前决定要不要定义它，就能实现不同的效果。

　　如果config.h存在的话，编译时Makefile会把宏HAVE\_CONFIG\_H传给编译器，所以如果没有定义HAVE\_CONFIG\_H 的话，我们的程序不应该把config.h include进去。

三、制作流程

请按照以下的执行顺序一步一步做：

第一步　编写configure.in

　　生成configure.in的方法有两个，一个是自己从零开始写，另一个方法是用autoscan，执行autoscan后会生成configure.scan，其中包含了一些模板内容，使用时只要把名字改成.in就可以。

　　configure.in中使用的命令有两种，一种是以AC开头，表示是由autoconf提供，另一种是以AM开头，代表由automake提供。

　　在configure.in我们可以完成很多检测动作，比如检查编译所需的程式、头文件、库等等，总之功能是十分强大，不过我们这里只检测了编译器和头文件，详细用法请看 GNU Manuals Online

　　以"dnl"为首的行为注释行(代码中绿色部份)。

configure.in

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. dnl 初始化autoconf，参数为入口函数所在的文件
2. AC\_INIT(src/tt.c)
4. dnl 初始化automake，参数为软件名称及版本号
5. AM\_INIT\_AUTOMAKE(tt, 1.0)
7. dnl 告诉automake我们所用的配置文件，一般为config.h
8. AM\_CONFIG\_HEADER(config.h)
10. dnl 这里是实现自定义参数的部份，见下面的说明
11. AC\_ARG\_ENABLE(popo, [ --enable-popo PoPo is present],,enable\_popo=no)
12. **if** test "$enable\_popo" = yes ; then
13. echo "PoPo is here!"
14. AC\_DEFINE(POPO)
15. **else**
16. echo "PoPo isn't here!"
17. fi
19. dnl 检测编译器
20. AC\_PROG\_CC
22. dnl 检测Standard C的头文件
23. AC\_HEADER\_STDC
25. dnl 输出文件，一般来说顶级目录和各子目录都应有Makefile输出
26. AC\_OUTPUT(Makefile src/Makefile)

　　./configure的自定义参数有两种，一种是开关式(--enable-XXX或--disable-XXX)，另一种是开放式，即后面要填入一串字符(--with-XXX=yyyy)参数。  
　　上述代码中用的是开关式，第一个参数是参数名，第二个是说明(执行"./configure --help"后所显示出来的内容)，最后一个参数是默认值。一般来说默认值和用户提示应该是互斥的，即默认值是no的话，应提示用户用enable进行修改，反之亦然。  
　　从上面的代码中可以看到，如果$enable\_popo为yes的话，就用AC\_DEFINE来定义POPO这个宏(Macro)，否则就不定义，我们在这里所使用到的宏，一定要在acconfig.h中声明。

第二步　运行aclocal　在tt目录下运行aclocal，将会生成aclocal.m4。

第三步　编写acconfig.h

　　在configure.in中使用到的宏(Macro)，都应该在这个文件声明，一般用#undef来声明。

acconfig.h

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. #undef POPO

第四步　运行autoheader

　　运行autoheader后会根据configure.in、acconfig.h和系统预设的acconfig.h来生成config.h.in。

第五步　编写Makefile.am

　　一般来说，在顶级目录和各子目录都应有一个Makefile.am。

Makefile.am

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. AUTOMAKE\_OPTIONS = foreign
2. SUBDIRS = src

第一行是告诉automake不要检测目录中是否存在AUTHORS、README等文件。  
第二行是告诉automake处理src这个子目录。

src/Makefile.am

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. AUTOMAKE\_OPTIONS = foreign
2. bin\_PROGRAMS = tt
3. tt\_SOURCES = tt.c qq.c qq.h

第一行作用同前。  
第二行是目标执行档的名称。  
第三行是生成tt这个执行档所需的所有源程序和头文件名称。

第六步　运行automake

　　接著可以执行automake了，在命令行下输入

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. automake -a　和 automake -a src/Makefile

　　使用"automake -a"或"automake --add-missing"，会自动将install.sh、mkinstalldirs等文件补齐，否则会出错，切记!

第七步　运行autoconf

　　最后，可以执行autoconf了，完成后将会生成最终的configure！

四、编译&测试

　　用默认值编译：

[root@chiosoft tt]# ./configure  
Checking for ......  
PoPo isn't here!  
Checking for ......

[root@chiosoft tt]# make  
......

[root@chiosoft tt]# src/tt  
Hello, I am teacher(23), pls tell me your names!  
My name is QQ

　　默认状态下，我们没有定义宏POPO，所以./configure时输出"PoPo isn't here!"，运行时也只有QQ来报到。

　　再看看这个：   
[root@chiosoft tt]# ./configure --help  
......  
--enable and --with options recognized:  
　--enable-popo PoPo is present

[root@chiosoft tt]# ./configure --enable-popo  
Checking for ......  
PoPo is here!  
Checking for ......

[root@chiosoft tt]# make  
......

[root@chiosoft tt]# src/tt  
Hello, I am teacher(23), pls tell me your names!  
My name is PoPo!  
My name is QQ  
QQ: Hey PoPo, long time no see.

　　可以看到./configure时输出"PoPo is here!"，执行结果也完全不一样！

　　此外，我们也可以用make install来安装，预设是安装至/usr/local/bin下，当然，这些都是可以修改的。

五、生成发布包tarball

　　好了，至今为止，我们的小软件已经测试完毕，可以发布了，在tt下有很多文件，有的是我们自己写的，也有些是编译时生成的临时档案，到底哪些需 要打包到发行包中呢？当然你可以自己一个一个文件挑选，但用automake生成的Makefile提供了几个极方便的功能给我们。

　　我们可以用make dist或make distcheck来生成相应的tarball，其中后者还会帮我们测试发布包能否正常工作，所以个人推荐使用make distcheck。

　　看到了吧？发布包tt-0.1.0.tar.gz已经放到tt下了，有没有留意，这里用的软件名及版本号正是 configure.in中AM\_INIT\_AUTOMAKE所带的两个参数！现在你可以试试把它解压安装了。

附上操作命令：

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. aclocal
2. autoheader
3. automake
4. automake -a　和 automake -a src/Makefile
5. autoconf

**[cpp]** [view plain](https://blog.csdn.net/earbao/article/details/52153600) [copy](https://blog.csdn.net/earbao/article/details/52153600)

1. ./configure
2. make
3. src/tt
4. ./configure --help
6. ./configure --enable-popo
7. make
8. src/tt
9. make distcheck

源码下载：

