**Makefile.am 规则和实例详解**

[](https://www.jianshu.com/u/b1d20cb66d89)

[id被吃](https://www.jianshu.com/u/b1d20cb66d89)

2017.02.07 20:30\* 字数 3013 阅读 2013评论 0喜欢 2

转载自：[Makefile.am 规则和实例详解--鹏博客](https://link.jianshu.com?t=http:/www.ivpeng.com/pblog/makefile-am.html)

编写Linux C 程序的时候，自己来写Makefile着实的让人很头疼，如果是简单的项目自己写写也就罢了，但是如果遇到大项目自己写Makefile，那是要弄死人的，所以最近在研究Autotools工具自动生成Makefile，在用到autotools工具生成Makefile的时候，还是有一部分需要自己来完成的，那就是Makefile.am文件。

项目中写在源文件里的Makefile.am是一种比我们了解的Makefile更高层次的编译规则，它可以和编写的configure.in（了解更多configure.in的规则请阅读《[configure.ac (configure.in)详解](https://link.jianshu.com?t=http:/www.ivpeng.com/pblog/configure-ac.html)》）文件一起通过调用automake命令，来生成Makefile.in文件，然后再调用./configure，将Makefile.in文件自动的生成Makefile文件。所以Makefile.am文件是要自动生成Makefile必不可少的元素，下面鹏博客就来和大家着重的学习下Makefile.am的写法和规则。

先来说下Makefile.am中常见的文件编译类型，详细的编译类型和全局变量鹏博客会在下面在图表中列出：

PROGRAMS           表示可执行文件

SOURCES              表示源文件

HEADERS              头文件。

LIBRARIES             表示库文件

LTLIBRARIES         这也是表示库文件，前面的LT表示libtool。

DATA                       数据文件，不能执行。

SCRIPTS                脚本文件，这个可以被用于执行。如：example\_SCRIPTS，如果用这样的话，需要我们自己定义安装目录下的example目录，很容易的，往下看。

一、基本写法

下面就直接引入一个例子进行详细讲解，如下：

AUTOMAKE\_OPTIONS = foreign

bin\_PROGRAMS = client

client\_SOURCES = key.c connect.c client.c main.c session.c hash.c

client\_CPPFLAGS = -DCONFIG\_DIR=\“$(sysconfdir)\” -DLIBRARY\_DIR=\”$(pkglibdir)\”

client\_LDFLAGS = -export-dynamic -lmemcached

noinst\_HEADERS = client.h

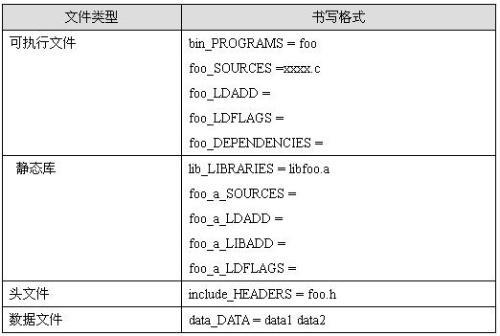
INCLUDES = -I/usr/local/libmemcached/include/

client\_LDADD = $(top\_builddir)/sx/libsession.la \

$(top\_builddir)/util/libutil.la

上面就是一个Makefile.am示例文件，这个文件是用于生成client可执行应用程序，引用了两个静态库和MC等动态库的连接。

先来看个图表一（列出了可执行文件、静态库、头文件和数据文件，四种书写Makefile.am文件个一般格式。）：



**对于可执行文件和静态库类型，如果只想编译，不想安装到系统中，可以用noinst\_PROGRAMS代替bin\_PROGRAMS，noinst\_LIBRARIES代替lib\_LIBRARIES。以此类推。**

根据这个图表一来分析下具体内容：

**AUTOMAKE\_OPTIONS**：这个是用来设定automake的选项。automake主要是帮助开发GNU软件的人员维护软件套件，一般在执行automake时会检查目录下是否存在标准GNU套件中应具备的文件档案，例如NEWS、AUTHOR、ChangeLog等，设成foreign时，automake会改用一般软件套件标准来检查，而gnu是缺省设置，该级别下将尽可能地检查包是否服从GNU标准，gnits是严格标准，不推荐。

**bin\_PROGRAMS**：表示要生成的可执行应用程序文件，这里的bin表示可执行文件在安装时需要被安装到系统中，如果只是想编译。不想被安装到系统中，可以用noinst\_PROGRAMS来代替。

那么整个第一行 bin\_PROGRAMS=client 详细表示什么意思那，解释如下：

PROGRAMS知道这是一个可执行文件。

client表示编译的目标文件。

bin表示目录文件被安装到系统的目录。

如程序和图片所示，包括头文件，静态库的定义等等都是这种形式，如lib\_LIBRARIES=util，表示将util库安装到lib目录下。

继续解释文件内容：

**client\_SOURCES**：表示生成可执行应用程序所用的所有源文件源文件，多个就空格隔开，我们注意到client\_是由前面的bin\_PROGRAMS指定的，如果前面是生成example, 那么这里也就变成example\_SOURCES，其它的规则类似标识也是一样。

**client\_CPPFLAGS**：这个和我们写Makefile的时候意思是一样的，都表示C语言的预处理器参数，这里指定了DCONFIG\_DIR，以后在程序中，就可以直接使用CONFIG\_DIR,不要把这个和另一个CFLAGS混淆，后者表示编译器参数。

**client\_LDFLAGS**：表示在连接时所需要的库文件选项标识。这个也就是对应一些如-l,-shared等选项。

**noinst\_HEADERS**：表示该头文件只是参加可执行文件的编译，而不用安装到安装目录下。如果需要安装到系统中，可以用include\_HEADERS来代替。

**INCLUDES**：表示连接时所需要的头文件。

**client\_LDADD**：表示连接时所需要的库文件,这里表示需要两个库文件的支持，下面会看到这个库文件又是怎么用Makefile.am文件后成的。

如图表二：



**全局变量**，可能有人注意到文件中的$(top\_builddir)等全局变量，其实这个是Makefile.am系统定义的一个基本路径变量，表示生成目标文件的最上层目录，如果这个Makefile.am文件变成其它的Makefile.am文件，那么这个就表示其它的目录，而不是这个当前目录。我们还可以使用$(top\_srcdir)，这个表示工程的最顶层目录，其实也是第一个Makefile.am的入口目录，因为Makefile.am文件可以被递归性的调用。

如图表三：（在Makefile.am中尽量使用相对路径，系统预定义了两个基本路径）



**$(sysconfdir)**：在系统安装工具的时候，我们经常能遇到配置安装路径的命令，如：./configure –prefix=/install/apache  其实在调用这个之后，就定义了一个变量$(prefix), 表示安装的路径，如果没有指定安装的路径，会被安装到默认的路径，一般都是/usr/local。在定义$(prefix)，还有一些预定义好的目录,其实这一些定义都可以在顶层的Makefile文件中可以看到，如下面一些值：

bindir = $(prefix)/bin。

libdir = $(prefix)/lib。

datadir=$(prefix)/share。

sysconfdir=$(prefix)/etc。

includedir=$(prefix)/include。

这些量还可以用于定义其它目录，例如我想**将client.h安装到include/client目录下**，这样写Makefile.am文件：

clientincludedir=$(includedir)/client

clientinclude\_HEADERS=$(top\_srcdir)/client/client.h

这就达到了我的目的，相当于定义了一个安装类型，这种安装类型是将文件安装到include/client目录下。

我们自己也可以**定义新的安装目录下的路径**，如我在应用中简单定义的：

devicedir = ${prefix}/device

device\_DATA = package

这样的话，package文件会作为数据文件安装到device目录之下，这样一个可执行文件就定义好了。注意，这也相当于定义了一种安装类型：devicedir，所以你想怎么安装就怎么安装，后面的XXXXXdir，dir是固定不变的。

二、配置静态库

下面我们来说下编译静态库和编译动态库，我们说下静态库，下面这个例子比较简单。直接指定 XXXX\_LTLIBRARIES或者XXXX\_LIBRARIES就可以了。同样如果不需要安装到系统，将XXXX换成noinst就可以。

一般推荐使用libtool库编译目标，因为automake包含libtool，这对于跨平台可移植的库来说，是一个很好的事情。

看例子如下：

noinst\_LTLIBRARIES = libutil.la

oinst\_HEADERS = inaddr.h util.h compat.h pool.h xhash.h url.h device.h

ibutil\_la\_SOURCES = access.c config.c datetime.c hex.c inaddr.c log.c device.c pool.c rate.c sha1.c stanza.c str.c xhash.c

ibutil\_la\_LIBADD = @LDFLAGS@

第一行的noinst\_LTLIBRARIES，这里要注意的是LTLIBRARIES，另外还有LIBRARIES，两个都表示库文件。前者表示libtool库，用法上基本是一样的。如果需要安装到系统中的话，用lib\_LTLIBRARIES。

**.la**为libtool自动生成的一些共享库，vi编辑查看，主要记录了一些配置信息。可以用如下命令查看\*.la文件的格式   $file \*.la

**.a**为静态库,是好多个.o合在一起,用于静态连接

如果想编译 .a 文件，那么上面的配置就改成如下结果：

noinst\_LTLIBRARIES = libutil.a

oinst\_HEADERS = inaddr.h util.h compat.h pool.h xhash.h url.h device.h

ibutil\_a\_SOURCES = access.c config.c datetime.c hex.c inaddr.c log.c device.c pool.c rate.c sha1.c stanza.c str.c xhash.c

ibutil\_a\_LIBADD = @LDFLAGS@

注意：静态库编译连接时需要其它的库的话，采用XXXX\_LIBADD选项，而不是前面的XXXX\_LDADD。编译静态库是比较简单的，因为直接可以指定其类型。

三、配置动态库

如果想要编译XXX.so动态库文件，需要用到\_PROGRAMS类型，有一个关于安装路径的问题，如果希望将动态库安装到lib目录下，按照前面所说的，只需要写成lib\_PROGRAMS就可以了，lib表示安装的路径，但是automake不允许这样直接定义，所以可以采用下面的办法，同样是将动态库安装到lib目录下：

projectlibdir=$(libdir)//新建一个目录，就是该目录就是lib目录

projectlib\_PROGRAMS=project.so

project\_so\_SOURCES=xxx.C

project\_so\_LDFLAGS=-shared -fpic//GCC编译动态库的选项

这个动态库的编译写法是鹏博客网上总结的，希望有要的人自己来验证下。

四、SUBDIRS功能用法

**SUBDIRS**这是一个很重要的词，我们前面生成了一个目标文件，但是一个大型的工程项目是由许多个可执行文件和库文件组成，也就是包含多个目录，每个目录下都有用于生成该目录下的目标文件的Makefile.am文件，但顶层目录是如何调用，才能使下面各个目录分别生成自己的目标文件呢？就是SUBDIRS关键词的用法了。

看一下我的工程项目，这是顶层的Makefile.am文件

EXTRA\_DIST = Doxyfile.in README.win32 README.protocol contrib UPGRADE

devicedir = ${prefix}/device

device\_DATA = package

SUBDIRS = etc man

ifUSE\_LIBSUBST

SUBDIRS += subst

endif

SUBDIRS += tools io sessions util client dispatch server hash storage sms

SUBDIRS表示在处理目录之前，要递归处理哪些子目录，要注意处理的顺序。比如配置中的client对sessions和utils这两上目标文件有依赖关系，就在client之前需要处理这两个目标文件。

**EXTRA\_DIST**：将哪些文件一起打包。

五、打包处理

**Automake会自动的打包**，自动打包的内容如下：

所有程序的源文件。

所有子目录里的的Makefile.am文件。

Makefile.am中包含的文件。

./configure所要读取的文件。

EXTRA\_DIST所指定的文件。

dist和nodist指定的文件，也可将其中一个源文件指定为不打包：

例如： nodist\_client\_SOURCES = client.c

六、最后

这里是鹏博客总结的一些比较实用的Makefile.am的写法和规则，看完了这篇文章已经可以很详细的理解这个文件的内容，写起来也应该不会陌生，但automake还有许多其他的规则需要掌握，鹏博客将会继续全面的总结关于autotools 的一些规则和写法，希望对大家有用处。也欢迎大家指出问题，帮我完善这个博客，希望大家支持！

[automake的Makefile.am](https://link.jianshu.com?t=http:/www.ivpeng.com/ptag/automake%e7%9a%84makefile-am)[Makefile.am写法](https://link.jianshu.com?t=http:/www.ivpeng.com/ptag/makefile-am%e5%86%99%e6%b3%95)