欣宇的专栏

Makefile.am文件的实例讲解

分类： Linux+C 2012-09-27 11:22 2302人阅读 评论(1) 收藏 举报

makefile include 编译器 跨平台 工作 tools

Makefile.am是一种比Makefile更高层次的编译规则，可以和configure.in文件一起通过调用automake命令，生成Makefile.in文件，再调用./configure的时候，就将Makefile.in文件自动生成Makefile文件了。所以Makefile.am文件是比Makefile文件更高的抽象。

下面我根据自己的工作中的一些应用，来讨论Makefile.am的编写。我觉得主要是要注意的问题是将编译什么文件？这个文件会不会安装？这个文件被安装到什么目录下？可以将文件编译成可执行文件来安装，也可以编译成静态库文件安装，常见的文件编译类型有下面几种：

1. PROGRAMS。表示可执行文件
2. LIBRARIES。表示库文件
3. LTLIBRARIES。这也是表示库文件，前面的LT表示libtool。
4. HEADERS。头文件。
5. 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。
6. DATA。数据文件，不能执行。

一，可执行文件

先看一个实例：

```cpp
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
```

上面就是一个全部的Makefile.am文件，这个文件用于生成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。分析一下：

bin_PROGRAMS：表示指定要生成的可执行应用程序文件，这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。

一个简单的问题是：bin_PROGRAMS=client 这一行表示什么意思？解释如下：

1. PROGRAMS知道这是一个可执行文件。
2. client表示编译的目标文件。
3. bin表示目录文件被安装到系统的目录。

后面的包括头文件，静态库的定义都是这种形式，如lib_LIBRARIES=util，表示将util库安装到lib目录下。继续解释

C++类的函数重载 (1)
No module named yum (1)
WebRTC体系结构 (1)
GTK安装出错 (1)
Linux进程ID的内核管理 (1)
生产者/消费者 (1)
C语言的HashTable简单 (1)

推荐文章

最新评论

No module named yum 错误
yiran1924: 为什么我改了还是不行？
Linux进程ID的内核管理
zongmumask: 很详细，谢谢啦！
C++类的函数重载
幽冥落枫: 受教了！ C++提供using 语句可以解决这个问题：using Base::display; 编译...
GTK安装出错
liang0000zai: yum install libXext-devel 请问你这个libXext-devel包是在哪下载...
生产者/消费者
TonyJiang08: 你的这个不错，可以改造成多生产者和多消费者的例子，我运行成功了，没问题
Makefile.am文件的实例讲解
满衣兄: 通俗易懂
WebRTC体系结构
fov42550564: 难道是传说中的程欣宇老师~~~？
inet_ntoa在64位机器上出错
bo: 转载啦
日志模块的C语言实现
zmxiangde_88: @M_O_Bz:谢谢，下次再分享时，把源码帖上。
C语言的HashTable简单实现
M_O_Bz: 这，还是那个问题，代码组织的不明确，很多细节都没写清楚，能把xxx.c 和xxx.h摆出就好了。。。...

上面的文件

client_SOURCES：表示生成可执行应用程序所用的源文件，这里注意，client_是由前面的bin_PROGRAMS指定的，如果前面是生成example,那么这里就是example_SOURCES，其它的类似标识也是一样。

client_CPPFLAGS：这和Makefile文件中一样，表示C语言预处理器参数，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：这个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：这个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：连接时所需要的头文件。

client_LDADD：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。

再谈谈关于上文中的全局变量引用，可能有人注意到$(top_builddir)等全局变量（因为这个文件之前没有定义），其实这个变量是Makefile.am系统定义的一个基本路径变量，表示生成目标文件的最上层目录，如果这个Makefile.am文件被其它的Makefile.am文件，这个会表示其它的目录，而不是这个当前目录。还可以使用$(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。

下面再说一下上文中出现的$(sysconfdir)，在系统安装时，我们都记得先配置安装路径，如./configure --prefix=/install/apache 其实在调用这个之后，就定义了一个变量$(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：

```
[html]
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：

```
[cpp]
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，静态库文件

编译静态库和编译动态库是不一样的，我们先看静态库的例子，这个比较简单。直接指定 XXXX_LTLIBRARIES或者XXXX_LIBRARIES就可以了。如果不需要安装到系统，将XXXX换成noinst就可以。还是再罗嗦一下：

- 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，肯定是一个福音。

看例子如下：

```
[cpp]
```

```
noinst_LTLIBRARIES = libutil.la

noinst_HEADERS = inaddr.h util.h compat.h pool.h xhash.h url.h device.h

libutil_la_SOURCES = access.c config.c datetime.c hex.c inaddr.c log.c device.c pool.c rate.

libutil_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。

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，动态库文件

想要编译XXX.so文件，需要用_PROGRAMS类型，这里一个关于安装路径要注意的问题是，我们一般希望将动态库安装到lib目录下，按照前面所讨论的，只需要写成lib_PROGRAMS就可以了，因为前面的lib表示安装路径，但是automake不允许这么直接定义，可以采用下面的办法，也是将动态库安装到lib目录下

```
[cpp]
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
```

这个动态库我没有测试，在网上找的一些资料，然后整理的。

四，SUBDIRS的用法

这是一个很重要的关键词，我们前面生成了一个一个的目标文件，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

```
[cpp]
EXTRA_DIST = Doxyfile.in README.win32 README.protocol contrib UPGRADE

devicedir = ${prefix}/device
device_DATA = package

SUBDIRS = etc man
if USE_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，这里还要注意处理的顺序。比如我的client对sessions和utils这两上目标文件有依赖，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，关于打包

Automake会自动的打包，自动打包的内容如下：

1. 所有源文件。
2. 所有的Makefile.am文件。
3. configure读取的文件。
4. Makefile.am中包含的文件。
5. EXTRA_DIST指定的文件。
6. 采用dist及nodist指定的文件，如可以将某一源文件指定为不打包：
   nodist_client_SOURCES = client.c

六，完成之前

基本上我介绍的都是我实际工作中所用的实例，GNU automake工具包含的东西，完不止这些。我介绍的过程也是我所认识的和参考的一些我所理解的，欢迎大家指出不足。

更多 0

上一篇 Linux中find命令使用

下一篇 C语言的HashTable简单实现