###安装m4

wget http://mirrors.kernel.org/gnu/m4/m4-1.4.13.tar.gz \

&& tar -xzvf m4-1.4.13.tar.gz \

&& cd m4-1.4.13 \

&& ./configure --prefix=/usr/local

make && make install

cd ..

#安装Autoconf:

wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz

tar xf autoconf\*

cd autoconf-2.69

sh configure --prefix /usr/local

make install

#安装Automake:

wget http://ftp.gnu.org/gnu/automake/automake-1.15.tar.gz

tar xf automake\*

cd automake-1.15

sh configure --prefix /usr/local

make install

#安装Libtool:

wget http://mirror.jre655.com/GNU/libtool/libtool-2.4.6.tar.gz

tar xf libtool\*

cd libtool-2.4.6

sh configure --prefix /usr/local

make install

关于Autotools

我们前面的章节中已经讲到了Makefile的使用（点击进入查看文章）。我们知道在Linux下面如果编译一个比较大型的项目，我们可以通过Makefile的方式来完成。

但是，我们又蛋疼了，Makefile拥有复杂的语法结构，甚至让人难以领会，当我们项目非常大的时候，维护Makefile会成为一件非常头疼的事情。于是我们就有了autotools工具，专门用来生成Makefile，这个工具让我们很大程度的降低了开发的难度。

Autotools并不是一个工具，而是一系列工具：

1. autoscan

2. aclocal

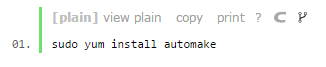
3. autoconf

4. autoheader

5. automake

记住，这一系列工具看着复杂，最终的目标还是生成Makefile

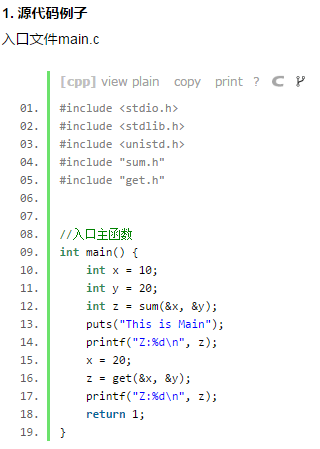
一般情况下系统中都会默认安装这一系列工具，如果未安装，则在Centeros中可以通过下面命令安装

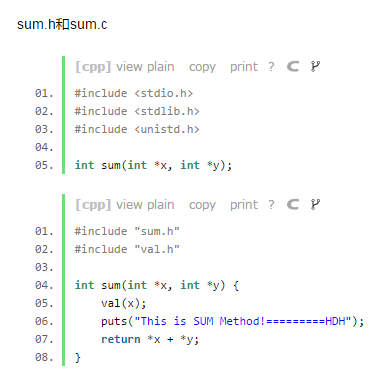


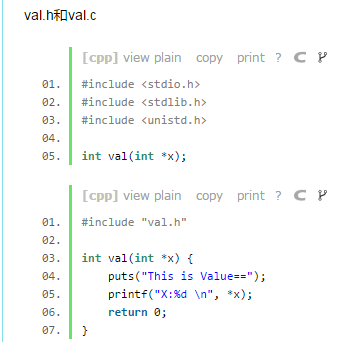
一、c源文件同一目录下Autotools的使用

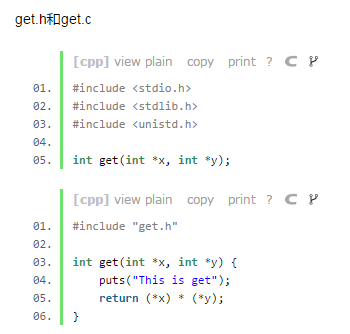
如果你的源文件都放在同一个目录下面，那么使用Autotools的时候会相对简单很多。比较著名的开源软件Memcache也是放在同一目录下的，你可以去看下它的源码包。

下面会按照步骤来实现同一目录下的Autotools工具的使用

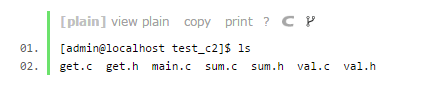






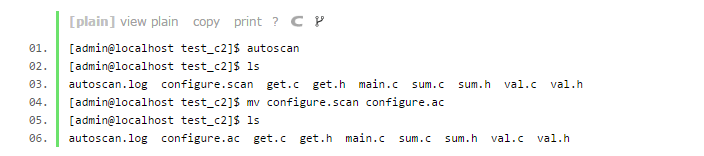


上面这个例子，我们在Makefile这篇文章中已经讲解过如何来手工编写Makefile编译。这边的话我们继续使用这个例子，实现Autotools的工具编译。

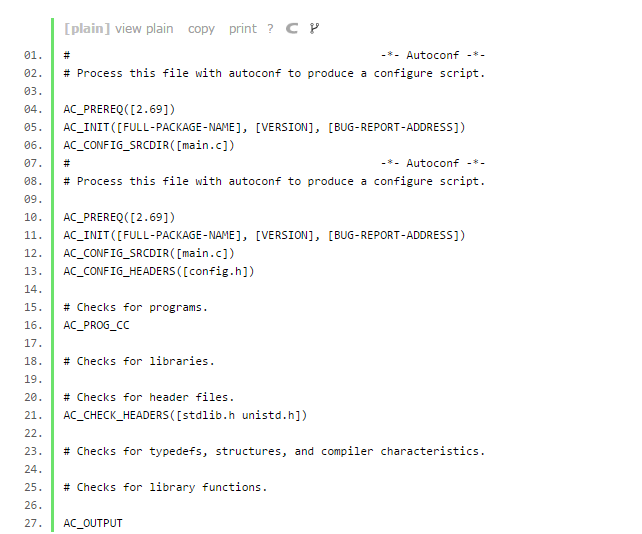


2. Autoscan命令

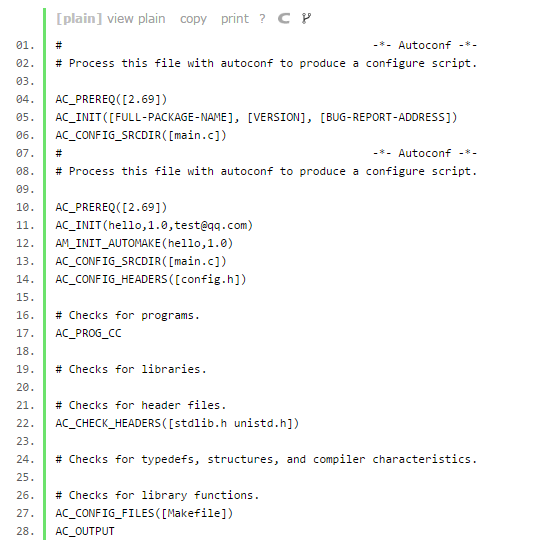
第一步，我们需要在我们的项目目录下执行autoscan命令。这个命令主要用于扫描工作目录，并且生成configure.scan文件。并且configure.scan需要重命令成configure.ac，然后编辑这个配置，我们才能继续执行后面的命令。



我们需要编辑configure.ac文件，首先我们打开configure.ac文件：



我们修改成：

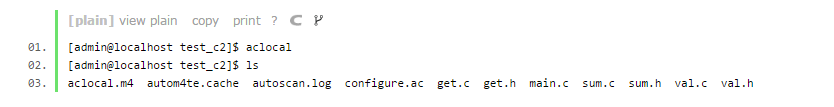


configure.ac标签说明：



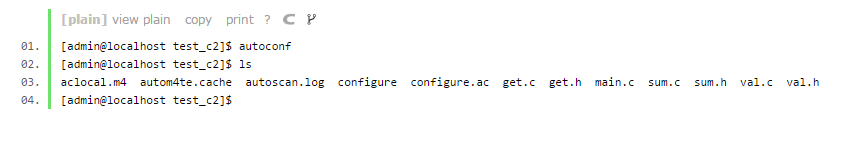
3. Aclocal命令

第二步，执行aclocal命令。扫描 configure.ac 文件生成 aclocal.m4文件, 该文件主要处理本地的宏定义，它根据已经安装的宏、用户定义宏和 acinclude.m4 文件中的宏将 configure.ac 文件需要的宏集中定义到文件 aclocal.m4 中。



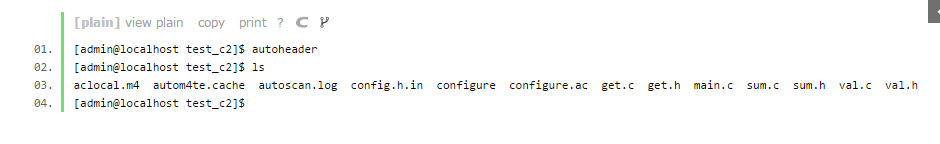
#### 4. Autoconf命令

第三步，执行autoconf命令。这个命令将 configure.ac 文件中的宏展开，生成 configure 脚本。这个过程可能要用到aclocal.m4中定义的宏。



5. Autoheader命令

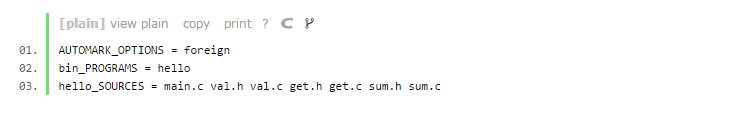
第四步，执行autoheader命令。该命令生成 config.h.in 文件。该命令通常会从 "acconfig.h” 文件中复制用户附加的符号定义。该例子中没有附加的符号定义, 所以不需要创建 "acconfig.h” 文件。



6. 创建Makefile.am文件

第五步，创建Makefile.am文件。Automake工具会根据 configure.in 中的参量把 Makefile.am 转换成 Makefile.in 文件。最终通过Makefile.in生成Makefile文件，所以Makefile.am这个文件非常重要，定义了一些生成Makefile的规则

Makefile.am：



1. AUTOMAKE\_OPTIONS：由于GNU对自己发布的软件有严格的规范, 比如必须附带许可证声明文件COPYING等，否则automake执行时会报错. automake提供了3中软件等级:foreign, gnu和gnits, 供用户选择。默认级别是gnu. 在本例中， 使用了foreign等级, 它只检测必须的文件。

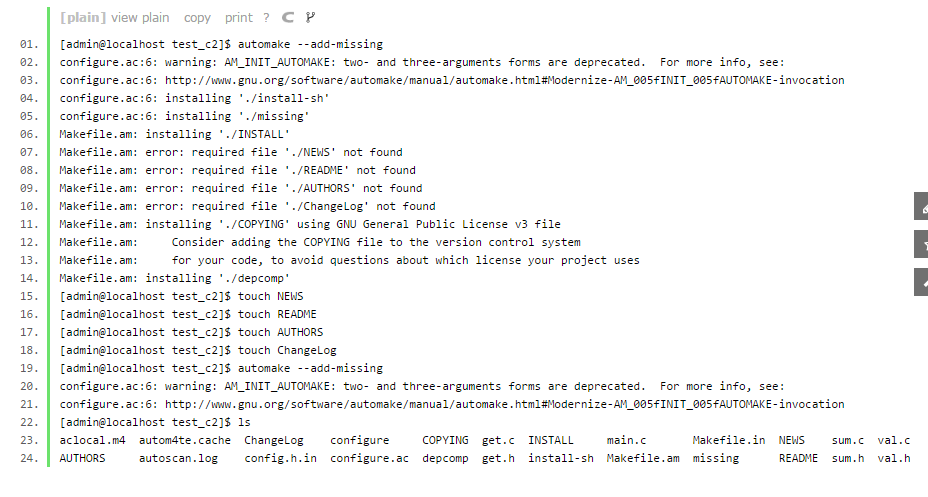
2. bin\_PROGRAMS = hello ：生成的可执行文件名称，生成多个可执行文件，可以用空格隔开。

3. hello\_SOURCES：生成可执行文件hello需要依赖的源文件。其中hello\_为可执行文件的名称。

具体Makefile.am后面我们会有一个章节专门讲这块内容。

7. Automake命令

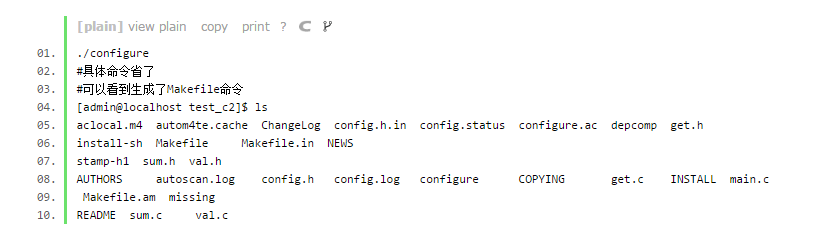
第六步，执行automake --add-missing命令。该命令生成 Makefile.in 文件。使用选项 "--add-missing" 可以让 Automake 自动添加一些必需的脚本文件。如果发现一些文件不存在，可以通过手工 touch命令创建。



8. configure命令

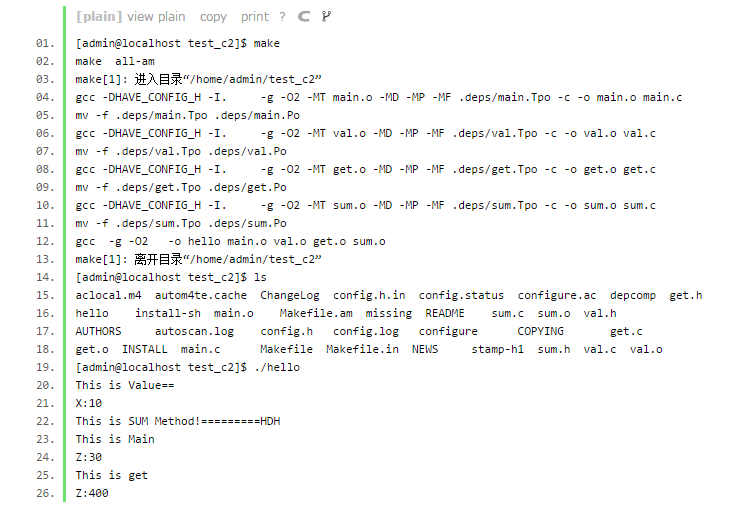
第七步，估计大家都对 ./congigure这个命令很熟悉吧。大部分linux软件安装都先需要执行./congigure，然后执行make和make install命令。

./congigure主要把 Makefile.in 变成最终的 Makefile 文件。configure会把一些配置参数配置到Makefile文件里面。



9. make命令

第八步，执行make命令，执行make命令后，就生成了可执行文件hello。



1. **c源文件不同目录下Autotools的使用**

**如果你的入口文件main.c和依赖的文件不是在同一个目录中的，使用Autotools来管理项目的时候会稍微复杂一下。**

**在不同的目录下，项目会生成\*.a文件的静态连接（静态连接相当于将多个.o目标文件合成一个）。最外层的main.c会通过静态连接方式来实现连接。**

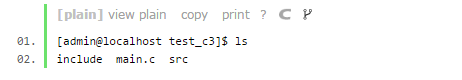
**1. 源代码例子**

**这个例子中会加入libevent和pthread，让例子稍显复杂，这样可以详细的介绍不同目录下的Autotools的使用。**

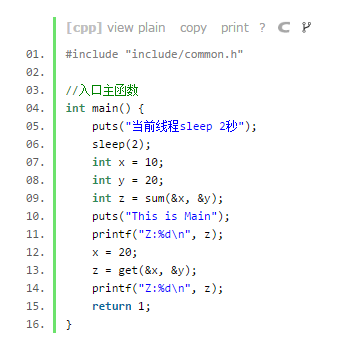
**我们创建两个目录：**

**include/ ：放置.h头文件**

**src/ ：放置.c 源文件**



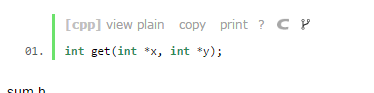
入口文件main.c：



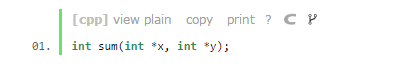
common.h文件：



get.h：



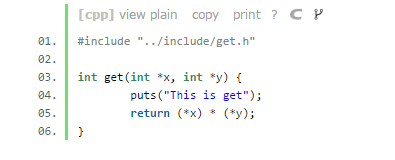
Sum.h



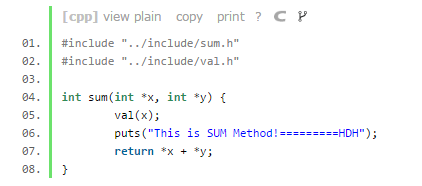
Val.h



Get.c



Sum.c

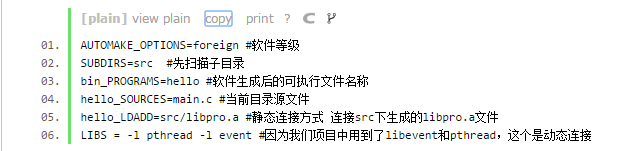


Val.c

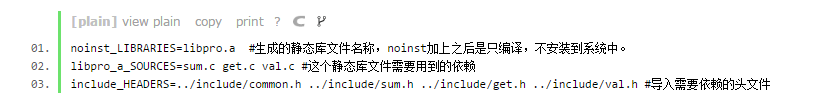


2. 创建Makefile.am文件

在项目根目录下先创建Makefile.am文件。



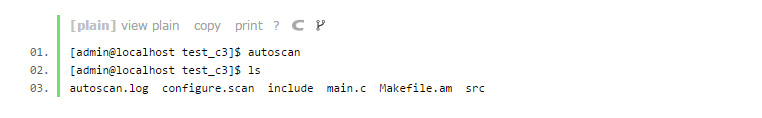
在src/目录下创建Makefile.am文件



说明：src/目录下面不加include\_HEADERS也是可以运行的，但是在使用make dist打包命令后，并不会将include/文件夹打包进去，所以还是需要加上include\_HEADERS。

1. 执行Autoscan命令

第一步，我们需要在我们的项目目录下执行autoscan命令。这个命令主要用于扫描工作目录，并且生成configure.scan文件。并且configure.scan需要重命令成configure.ac，然后编辑这个配置，我们才能继续执行后面的命令。

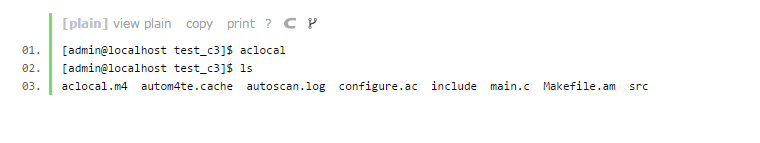


修改configure.ac文件，主要添加AC\_PROG\_RANLIB（生成静态库）；AC\_PROG\_LIBTOOL （用来生成动态库）



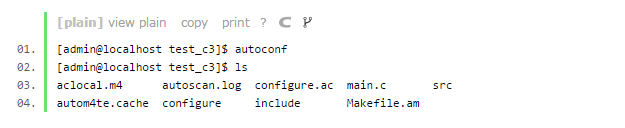
4. Aclocal命令

第二步，执行aclocal命令。扫描 configure.ac 文件生成 aclocal.m4文件, 该文件主要处理本地的宏定义，它根据已经安装的宏、用户定义宏和 acinclude.m4 文件中的宏将 configure.ac 文件需要的宏集中定义到文件 aclocal.m4 中。



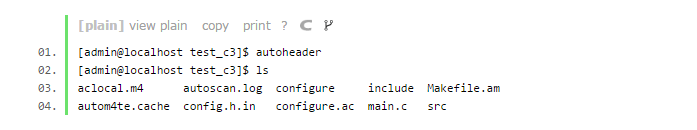
5. Autoconf命令

第三步，执行autoconf命令。这个命令将 configure.ac 文件中的宏展开，生成 configure 脚本。这个过程可能要用到aclocal.m4中定义的宏。



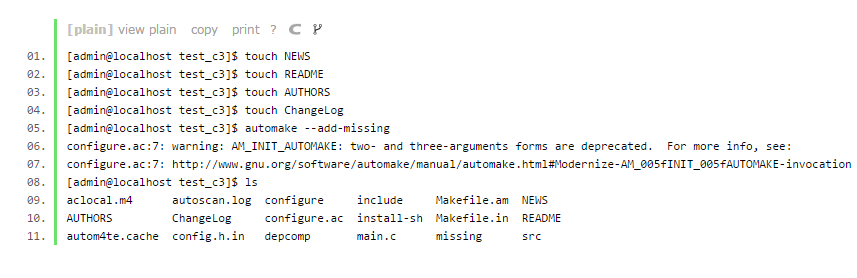
6. Autoheader命令

第四步，执行autoheader命令。该命令生成 config.h.in 文件。该命令通常会从 "acconfig.h” 文件中复制用户附加的符号定义。该例子中没有附加的符号定义, 所以不需要创建 "acconfig.h” 文件。



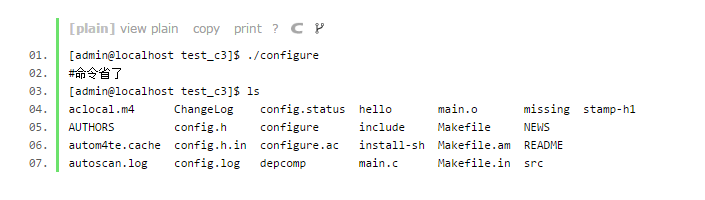
7. Automake命令

第五步，执行automake --add-missing命令。该命令生成 Makefile.in 文件。使用选项 "--add-missing" 可以让 Automake 自动添加一些必需的脚本文件。如果发现一些文件不存在，可以通过手工 touch命令创建。



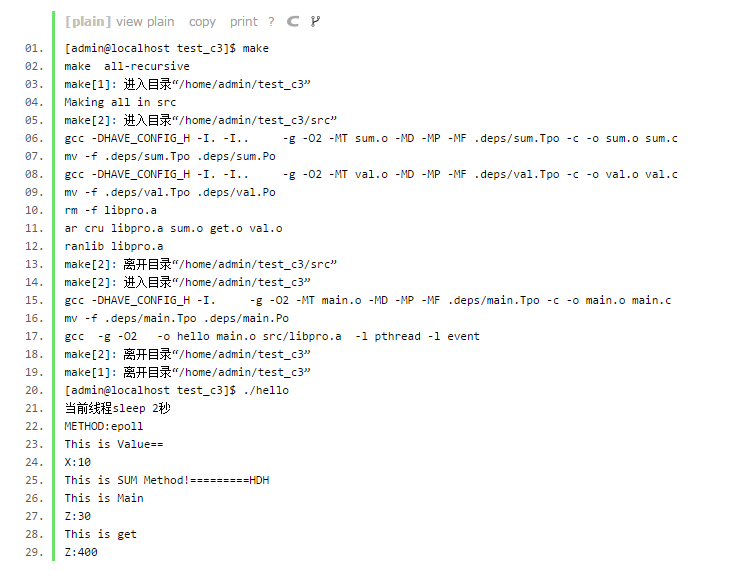
8. configure命令

第六步，执行./configure命令。./congigure主要把 Makefile.in 变成最终的 Makefile 文件。configure会把一些配置参数配置到Makefile文件里面。



9. make命令

第七步，执行make命令。make执行后，会生成hello的可执行文件。



### Autotools运行流程

#### 流程总结：

1. 执行**autoscan**命令。这个命令主要用于扫描工作目录，并且生成**configure.scan文件**。

2. 修改configure.scan为**configure.ac**文件，并且修改配置内容。

3. 执行**aclocal**命令。扫描 configure.ac 文件生成 aclocal.m4文件。

4. 执行**autoconf**命令。这个命令将 configure.ac 文件中的宏展开，生成 configure 脚本。

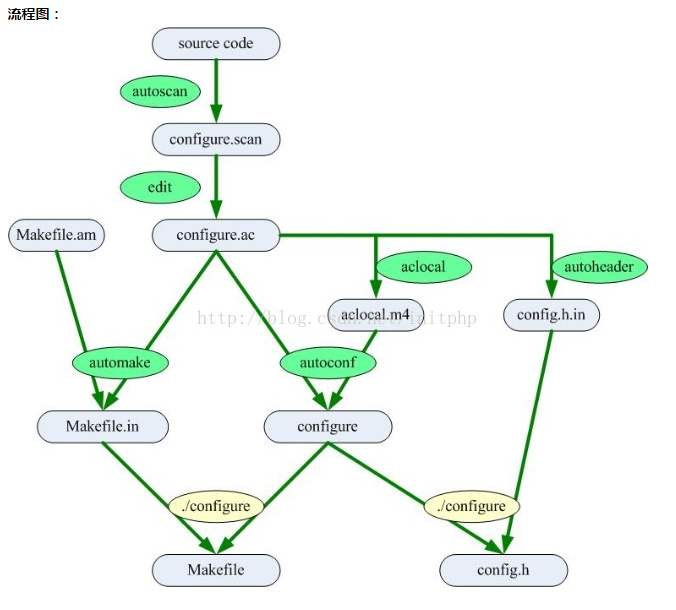
5. 执行**autoheader**命令。该命令生成 config.h.in 文件。

6. 新增Makefile.am文件，修改配置内容

7. 执行**automake** **--add-missing**命令。该命令生成 Makefile.in 文件。

8. 执行 **./congigure**命令。将Makefile.in命令生成Makefile文件。

9. 执行**make**命令。生成可执行文件。



Make命令详解

1. make命令：编译文件。make命令主要通过Makefile文件生成可执行文件。

2. make clean命令。清楚编译的文件，包括目标文件\*.o和可执行文件

3. make install 命令把目标文件安装到系统中。默认安装到/usr/local/bin目录下面。

4. make uninstall 命令，把目标文件从系统中卸载。

5. make dist 命令，打包发布。

如何使用发布的文件：

1. 下载到“hello-1.0.tar.gz”压缩文档

2. 使用“ tar -zxvf hello-1.0.tar.gz ”命令解压

3. 使用 “./configure” 命令，主要是生成Makefile命令，已经一些配置初始化。

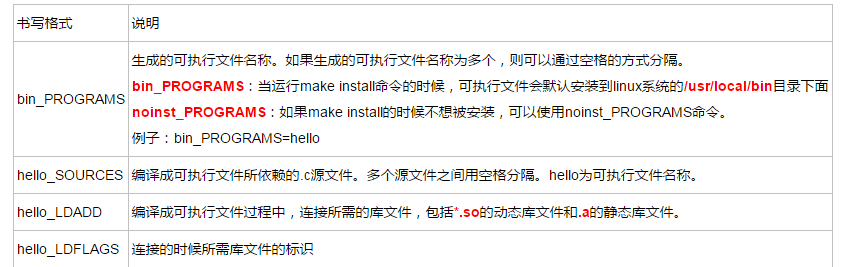
4. 使用 “make” 命令编译源代码文件生成软件包。

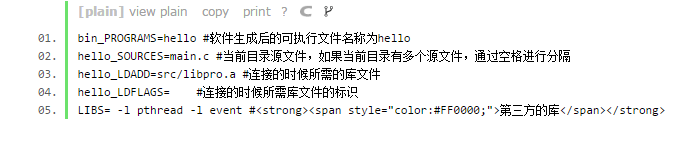
5. 使用“make install ”命令来安装编译后的软件包到系统中。

Makefile.am解读

1. 可执行文件类型

可执行文件类型主要是只最终生成的可执行的文件。例如我们上面“c源文件同一目录下Autotools的使用”中的例子。





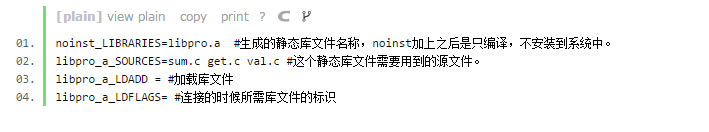
2. 静态库文件类型

静态库文件类型，一般会将c源码放在不同的文件夹中，并且每个文件夹中都会有各自的Makefile.am文件，并且会被编译成静态链接库 \*.a格式的文件。

如果对静态库和动态库还没有一个概念，可以看我的《Linux c 开发 - 静态库和动态库》

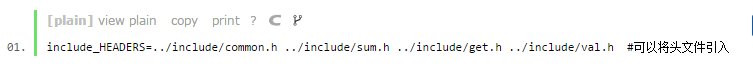
注意：静态库使用中，需要对configure.ac中加入AC\_PROG\_RANLIB





3. 头文件

我们一般需要导入一些\*.h的头文件，如果你在Makefile.am中没有标识需要导入的头文件，可能在make dist打包的时候出现问题，头文件可能不会被打进包里面。

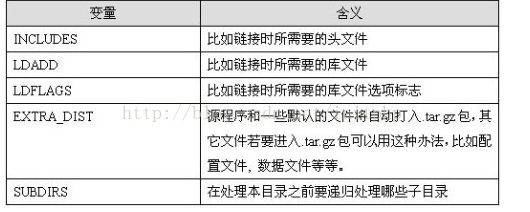


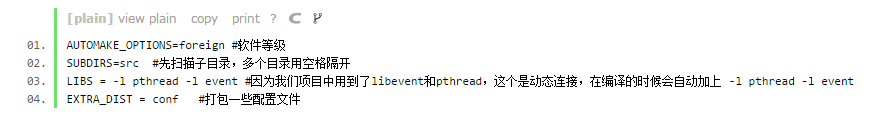
make install，头文件默认会被安装到linux系统/usr/local/include

#### 4. 数据文件



1. 常用变量



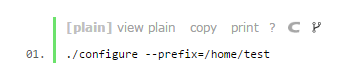


#### 6. 安装目录

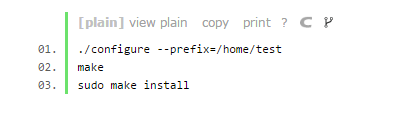
我们知道，默认情况下，执行make install命令，则会将文件安装到/usr/local/bin   /usr/local/include  /usr/local/lib目录下面。

我们可以通过命令**./configure --prefix=** 生成Makefile文件的时候，配置make install命令执行的时候的文件安装路径。

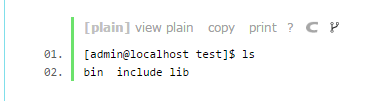
下面这个例子，我们在执行make install的时候，程序会被安装到/home/test目录下面。



执行下面一系列命令：

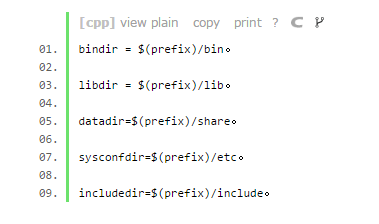


我们可以进入/home/test目录下看到相应的bin文件已经生成：



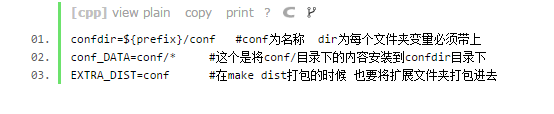
下面这些变量是已经定义好的安装路径的变量。

用户也可以修改这些变量。例如将bindir修改成$(prefix)/bin2



假如我们有自定义的文件夹，我们需要将这个文件夹下的内容安装到安装目录，

则需要配置一个自定义的文件夹目录confdir



confdir为需要创建的文件夹目录。

conf\_DATA为需要拷贝的文件内容到${prefix}/conf目录中去