

# 逍遥剑客的专栏 scucj

目录视图 摘要视图 RSS 订阅

## 个人资料



scucj

访问：415789次  
积分：4324  
等级：BLOG > 5  
排名：第5630名  
  
原创：73篇 转载：6篇  
译文：0篇 评论：178条

## 文章搜索

## 文章分类

- ASP.NET (9)
- C# (14)
- C/C++ (8)
- Database (3)
- Java (1)
- Other (9)
- UML (0)
- UNIX (3)
- Web Service (0)
- Windows编程 (2)
- XML (0)
- 心情随笔 (0)
- 数据结构与算法 (2)
- 计算机网络 (0)
- 设计模式 (24)
- 软件工程 (3)

## 文章存档

- 2012年04月 (1)
- 2011年12月 (1)
- 2010年12月 (5)
- 2010年07月 (1)
- 2010年05月 (1)

展开

## GNU Autotools的使用方法

标签：[makefile](#) [include](#) [compiler](#) [file](#) [gcc](#) [header](#)

2010-12-16 02:03 20614人阅读 评论(4) 收藏 举报

分类：[UNIX \(2\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

手工写Makefile是一件很有趣的事情，对于比较大型的项目，如果有工具可以代劳，自然是一件好事。在Linux系统开发环境中，GNU Autotools 无疑就充当了这个重要角色。(在Windows系统的开发环境中，IDE 如 Visual Studio，来管理项目也很方便。)

本文以一个简单项目为例子，来讲述GNU Autotools的一系列工具及其命令的用法。

autotools是系列工具，它主要由autoconf、automake、perl语言环境和m4等组成；所包含的命令有五个：

- (1) aclocal
- (2) autoscan
- (3) autoconf
- (4) autoheader
- (5) automake

### 一、准备源代码

(1) 目录project包含一个main.c的文件和两个子目录lib与include；lib目录中包含一个test.c，include目录中包含一个test.h。在系统中，显示如下：

```
[c-sharp]
01. [root@localhost project]# ls
02. include lib main.c
03. [root@localhost project]#
04. [root@localhost project]# ls include/
05. test.h
06. [root@localhost project]# ls lib/
07. test.c
08. [root@localhost project]#
```

(2)源代码如下：

```
[cpp]
01. /* project/main.c */
02. #include <stdio.h>
03. #include "include/test.h"
04. int main()
05. {
06.     printf("main entrance.\n");
07.     test_method();
08.     return 0;
```

## 阅读排行

MP算法和OMP算法及其	
PC-Lint的使用方法(一)	(77139)
在多线程中使用静态方法	(33818)
在 C# 中, (int), Int32.F	(25042)
GNU Autotools的使用方	(21214)
蚁群算法的源代码	(20606)
CUnit的用法	(20247)
多线程中对全局变量(整型	(14312)
动态修改WCF客户端配置	(9561)
OCX和DLL的区别	(9097)
	(9048)

## 评论排行

MP算法和OMP算法及其	(69)
蚁群算法的源代码	(40)
在多线程中使用静态方法	(11)
在 C# 中, (int), Int32.F	(11)
PC-Lint的使用方法(一)	(11)
CUnit的用法	(7)
动态增加和删除多个web	(7)
GNU Autotools的使用方	(4)
多线程中对全局变量(整型	(4)
在C#的事件、多播中使用	(3)

## 推荐文章

- \* 而立之年——三线程序员  
的年终告白
- \* Java集合框架中隐藏的设计  
套路
- \* Python脚本下载今日头条视频  
(附加Android版本辅助下载器)
- \* 人工智能的冷思考
- \* React Native 实战系列教程之  
热更新原理分析与实现

## 最新评论

- MP算法和OMP算法及其思想  
muranyufenzhe: @u011432163:  
同样也感觉不是OMP的, 你有木  
有啥想法, 可以一切交流一下  
啊!!
- 蚁群算法的源代码  
无奈的小心酸: 谢谢 帮大忙了
- MP算法和OMP算法及其思想  
marrymify2016: 博主大赞!
- MP算法和OMP算法及其思想  
萧神ZF: @u011432163:请问您得  
到答案了吗? 我也有同样的疑  
惑、
- MP算法和OMP算法及其思想  
lanchaoxi2011: 非常感谢博主的  
慷慨讲解。但是还有两点疑问希  
望得到您的解答。1. 对于文献  
【2】所附的算法, 我没有找到  
保...
- MP算法和OMP算法及其思想  
lanchaoxi2011: @nnxtby168:确  
实感觉附带的中文算法是MP而不  
是OMP算法, 因为看不出来算法  
里面哪里能保证新...
- MP算法和OMP算法及其思想  
僵尸蛀虫: 博主, 我想行了行时

```
09. | }
```

### [c-sharp]

```
01. /* project/lib/test.c */
02. #include <stdio.h>
03. #include "../include/test.h"
04. void test_method()
05. {
06.     printf("test method.\n");
07. }
```

### [cpp]

```
01. /* project/include/test.h*/
02. void test_method();
```

## 二、autotools 使用步骤

2.1 使用autoscan命令, 它将扫描工作目录, 生成 configure.scan 文件。

### [c-sharp]

```
01. [root@localhost project]# autoscan
02. autom4te: configure.ac: no such file or directory
03. autoscan: /usr/bin/autom4te failed with exit status: 1
04. [root@localhost project]# ls
05. autoscan.log  configure.scan  include  lib  main.c
06. [root@localhost project]#
```

2.2 将configure.scan 文件重命名为configure.ac, 并做适当的修改。在 configure.ac 中, # 号开始的行是注释, 其他都是m4 宏命令; configure.ac里面的宏的主要作用是侦测系统。

### [c-sharp]

```
01. [root@localhost project]mv configure.scan configure.ac
02. [root@localhost project]# ls
03. autoscan.log  configure.ac  include  lib  main.c
04. [root@localhost project]#
05. [root@localhost project]# cat configure.ac
06. #                                     -*- Autoconf -*-
07. # Process this file with autoconf to produce a configure script.
08. AC_PREREQ(2.59)
09. AC_INIT(FULL-PACKAGE-NAME, VERSION, BUG-REPORT-ADDRESS)
10. AC_CONFIG_SRCDIR([main.c])
11. AC_CONFIG_HEADER([config.h])
12. # Checks for programs.
13. AC_PROG_CC
14. # Checks for libraries.
15. # Checks for header files.
16. # Checks for typedefs, structures, and compiler characteristics.
17. # Checks for library functions.
18. AC_OUTPUT
19. [root@localhost project]#
```

2.3 对 configure.ac 文件做适当的修改, 修改显示如下[1]:

### [c-sharp]

```
01. [root@localhost project]# cat configure.ac
02. #                                     -*- Autoconf -*-
03. # Process this file with autoconf to produce a configure script.
04. AC_PREREQ(2.59)
05. #AC_INIT(FULL-PACKAGE-NAME, VERSION, BUG-REPORT-ADDRESS)
06. AC_INIT(hello,1.0,abc@126.com)
07. AM_INIT_AUTOMAKE(hello,1.0)
08. AC_CONFIG_SRCDIR([main.c])
09. AC_CONFIG_HEADER([config.h])
10. # Checks for programs.
```

间, 还是没想通上面的OMP算法里面的求bn是可以保证原子与残差项是正交的, 后面使用...

在多线程中使用静态方法是否有? [lindexi\\_gd: 博主说的让我知道静态是安全, 如果不使用静态成员](#)

[MP算法和OMP算法及其思想](#)  
[STHBETTER: 2.1 倒数 第一个公式 Rnf 改为Xm; 第三个公式  \$Xr\(k+1\)\$  ...](#)

在C#中, (int), Int32.Parse() 和 [李中华11: 嗯, 来这里取取经](#)

```
11. AC_PROG_CC
12. # Checks for libraries.
13. # Checks for header files.
14. # Checks for typedefs, structures, and compiler characteristics.
15. # Checks for library functions.
16. AC_CONFIG_FILES([Makefile])
17. AC_OUTPUT
```

说明:

- (1) 以“#”号开始的行均为注释行。
- (2) AC\_PREREQ 宏声明本文要求的 autoconf 版本, 如本例中的版本 2.59。
- (3) AC\_INIT 宏用来定义软件的名称、版本等信息、作者的E-mail等。
- (4) AM\_INIT\_AUTOMAKE是通过手动添加的, 它是automake所必备的宏, FULL-PACKAGE-NAME是软件名称, VERSION是软件版本号。
- (5) AC\_CONFIG\_SCDIR 宏用来侦测所指定的源码文件是否存在, 来确定源码目录的有效性。此处为当前目录下main.c。
- (6) AC\_CONFIG\_HEADER 宏用于生成config.h文件, 以便 autoheader 命令使用。
- (7) AC\_PROG\_CC用来指定编译器, 如果不指定, 默认gcc。
- (8) AC\_OUTPUT 用来设定 configure 所要产生的文件, 如果是makefile, configure 会把它检查出 [市价的使用世 \](#) makefile.in文件产生合适的makefile。使用 Automake 时, 还需要一些其他的参数, 这些额外的宏用 [aclocal.m4](#) 生成。
- (9) AC\_CONFIG\_FILES宏用于生成相应的Makefile文件。

2.4 使用 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。[2]

```
[c-sharp]
01. [root@localhost project]# aclocal
02. [root@localhost project]# ls
03. aclocal.m4 autom4te.cache autoscan.log configure.in include lib main.c
04. [root@localhost project]#
```

2.5 使用 autoconf 命令生成 configure 文件。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。

```
[c-sharp]
01. [root@localhost project]# autoconf
02. [root@localhost project]# ls
03. aclocal.m4 autom4te.cache autoscan.log configure configure.in include lib main.c
```

2.6 使用 autoheader 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[2].

```
[cpp]
01. [root@localhost project]# autoheader
02. [root@localhost project]# ls
03. aclocal.m4 autom4te.cache autoscan.log config.h.in configure configure.in include lib mair
04. [root@localhost project]#
```

2.7 手工创建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。

```
[cpp]
01. [root@localhost project]# cat Makefile.am
02. AUTOMAKE_OPTIONS = foreign
03. bin_PROGRAMS = hello
04. hello_SOURCES = main.c include/test.h lib/test.c
```

说明:

(1) 其中的AUTOMAKE\_OPTIONS为设置automake的选项. 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定义要产生的执行文件名. 如果要产生多个执行文件, 每个文件名用空格隔开。

(3) hello\_SOURCES 定义"hello"这个可执行程序所需的原始文件. 如果"hello"这个程序是由多个源文件所产生的, 则必须把它所用到的所有源文件都列出来, 并用空格隔开. 如果要定义多个可执行程序, 那么需要对每个可执行程序建立对应的file\_SOURCES。

2.8 使用 Automake 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。

```
[cpp]
01. [root@localhost project]# automake --add-missing
02. configure.ac: installing `./install-sh'
03. configure.ac: installing `./missing'
04. Makefile.am: installing `./INSTALL'
05. Makefile.am: required file `./NEWS' not found
06. Makefile.am: required file `./README' not found
07. Makefile.am: required file `./AUTHORS' not found
08. Makefile.am: required file `./ChangeLog' not found
09. Makefile.am: installing `./COPYING'
10. Makefile.am: installing `./depcomp'
11. [root@localhost project]#
```

2.8.1 再次使用 automake ——add-missing 运行一次, 可以辅助生成几个必要的文件。

```
[cpp]
01. [root@localhost project]# automake --add-missing
02. Makefile.am: required file `./NEWS' not found
03. Makefile.am: required file `./README' not found
04. Makefile.am: required file `./AUTHORS' not found
05. Makefile.am: required file `./ChangeLog' not found
06. [root@localhost project]# ls
07. aclocal.m4 autom4te.cache autoscan.log config.h.in config.h.in~ configure configure.ac COP\
08. [root@localhost project]#
```

2.8.2 在当前目录创建上面未发现的四个文件, 并再次使用 automake ——add-missing 运行一次。

```
[c-sharp]
01. [root@localhost project]# touch NEWS
02. [root@localhost project]# touch README
03. [root@localhost project]# touch AUTHORS
04. [root@localhost project]# touch ChangeLog
05. [root@localhost project]#
06. [root@localhost project]# automake --add-missing
07. [root@localhost project]# ls
08. aclocal.m4 autom4te.cache ChangeLog config.h.in~ config.status configure.ac depcomp INST/
09. AUTHORS autoscan.log config.h.in config.log configure COPYING include insti
10. [root@localhost project]#
```

2.9 使用 configure 命令, 把 Makefile.in 变成最终的 Makefile 文件。

```
[c-sharp]
01. [root@localhost project]# ./configure
02. checking for a BSD-compatible install... /usr/bin/install -c
03. checking whether build environment is sane... yes
04. checking for gawk... gawk
05. checking whether make sets $(MAKE)... yes
06. checking for gcc... gcc
```

```
07. checking for C compiler default output file name... a.out
08. checking whether the C compiler works... yes
09. checking whether we are cross compiling... no
10. checking for suffix of executables...
11. checking for suffix of object files... o
12. checking whether we are using the GNU C compiler... yes
13. checking whether gcc accepts -g... yes
14. checking for gcc option to accept ANSI C... none needed
15. checking for style of include used by make... GNU
16. checking dependency style of gcc... gcc3
17. configure: creating ./config.status
18. config.status: creating Makefile
19. config.status: creating config.h
20. config.status: config.h is unchanged
21. config.status: executing depfiles commands
22. [root@localhost project]# ls
23. aclocal.m4 autom4te.cache ChangeLog config.h.in config.log configure COPYING hello
24. AUTHORS autoscan.log config.h config.h.in~ config.status configure.ac depcomp include
25. [root@localhost project]#
```

Makefile文件已经生成成功。

### 三、Makefile的用法

3.1 make 命令，用来编译代码，默认执行"make all"命令，可以看到生成了"hello"的可执行文件，

```
[c-sharp]
01. [root@localhost project]# make
02. make all-am
03. make[1]: Entering directory `/home/chenjie/project'
04. gcc -g -O2 -o hello main.o test.o
05. make[1]: Leaving directory `/home/chenjie/project'
06. [root@localhost project]#
07. [root@localhost project]# ls
08. aclocal.m4 autom4te.cache ChangeLog config.h.in config.log configure COPYING hello
09. AUTHORS autoscan.log config.h config.h.in~ config.status configure.ac depcomp include
10. [root@localhost project]#
```

3.2 make clean 命令清除编译时的obj文件，它与 make 命令是对应关系，一个是编译，一个清除编译的文件

3.3 运行"./hello"就能看到运行结果:

```
[c-sharp]
01. [root@localhost project]# ./hello
02. main entrance.
03. test method.
04. [root@localhost project]#
```

3.4 make install 命令把目标文件安装到系统中。这一，直接输入hello, 就可以看到程序的运行结果。

```
[c-sharp]
01. [root@localhost project]# make install
02. make[1]: Entering directory `/home/chenjie/project'
03. test -z "/usr/local/bin" || mkdir -p -- "/usr/local/bin"
04. /usr/bin/install -c 'hello' '/usr/local/bin/hello'
05. make[1]: Nothing to be done for `install-data-am'.
06. make[1]: Leaving directory `/home/chenjie/project'
07. [root@localhost project]#
08. [root@localhost project]# hello
09. main entrance.
```

```
10. test method.
11. [root@localhost project]#
```

3.5 make uninstall 命令把目标文件从系统中卸载。

3.6 make dist 命令将程序和相关的文档打包为一个压缩文档以供发布，在本例子中，生成的打包文件名为：hello-1.0.tar.gz。

```
[c-sharp]

01. [root@localhost project]# make dist
02. { test ! -d hello-1.0 || { find hello-1.0 -type d ! -perm -200 -exec chmod u+w {} ';' && rm -fr h
03. mkdir hello-1.0
04. find hello-1.0 -type d ! -perm -755 -exec chmod a+rx, <a href="http://lib.csdn.net/base
/go" class='replace_word' title="Go知识库" target='_blank' style='color:#df3434; font-
weight:bold;'>Go</a>+rx {} /; -o /
05. ! -type d ! -perm -444 -links 1 -exec chmod a+r {} /; -o /
06. ! -type d ! -perm -400 -exec chmod a+r {} /; -o /
07. ! -type d ! -perm -444 -exec /bin/sh /home/chenjie/project
/install-sh -c -m a+r {} {} /; /
08. || chmod -R a+r hello-1.0
09. tardir=hello-1.0 && /bin/sh /home/chenjie/project
/missing --run tar chof - "$tardir" | GZIP=-best gzip -c >hello-1.0.tar.gz
10. { test ! -d hello-1.0 || { find hello-1.0 -type d ! -perm -200 -exec chmod u+w {} ';' && rm -fr h
11. [root@localhost project]# ls
12. aclocal.m4 autom4te.cache ChangeLog config.h.in config.log configure COPYING hello
13. AUTHORS autoscan.log config.h config.h.in~ config.status configure.ac depcomp hello-
1.0.tar.gz INSTALL lib main.o Makefile.am missing README test.o
14. [root@localhost project]#
```

#### 四 如何使用已发布的压缩文档

4.1 下载到“hello-1.0.tar.gz”压缩文档

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

4.3 使用“./configure”命令，主要的作用是对即将安装的软件进行配置，检查当前的环境是否满足要安装软件的依赖关系。

4.4 使用“make”命令编译源代码文件生成软件包。

4.5 使用“make install”命令来安装编译后的软件包。

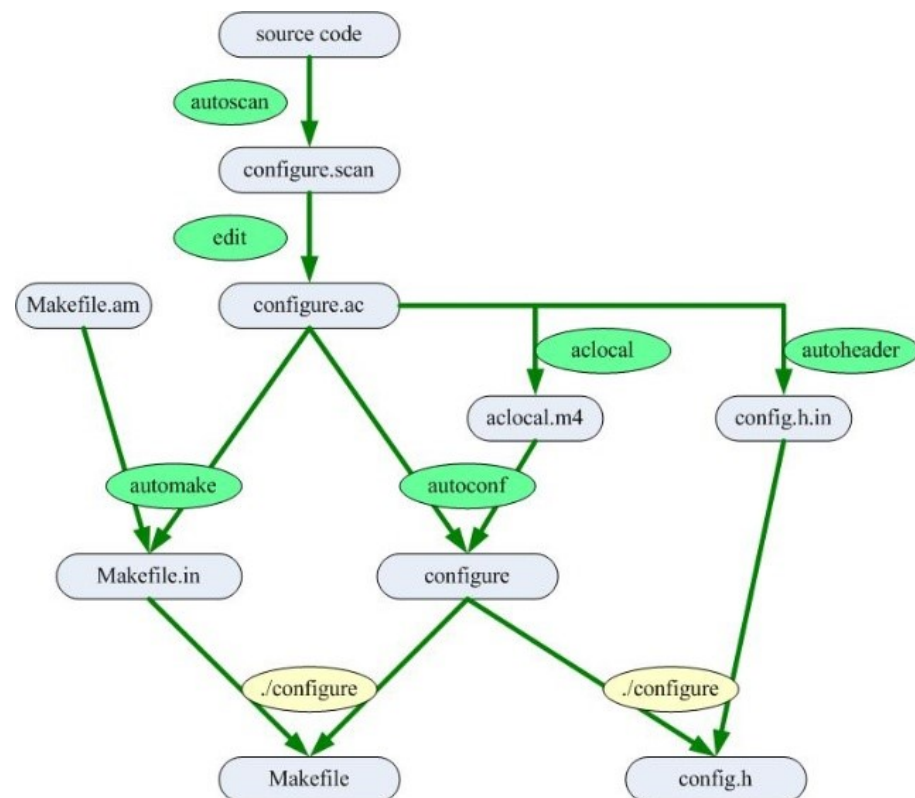
```
[c-sharp]

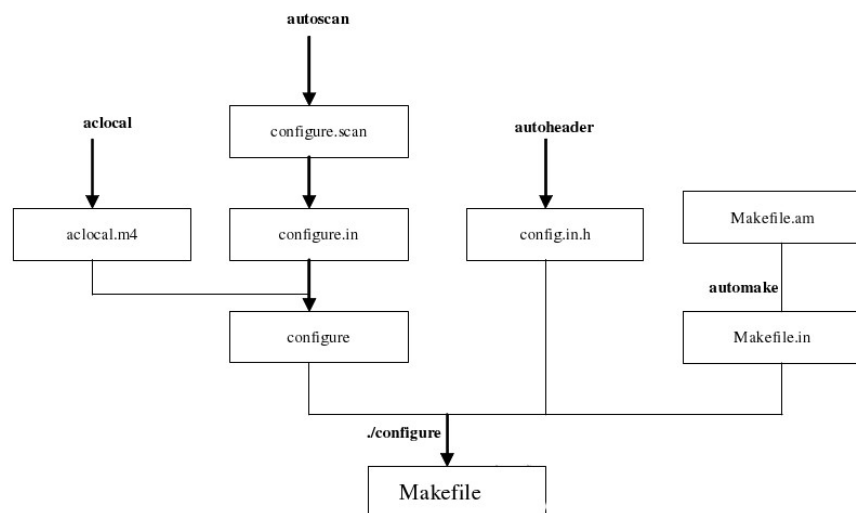
01. [root@localhost chenjie]# ls
02. hello-1.0.tar.gz
03. [root@localhost chenjie]# tar -zxvf hello-1.0.tar.gz
04. [root@localhost chenjie]# ls
05. hello-1.0 hello-1.0.tar.gz
06. [root@localhost chenjie]# cd hello-1.0
07. [root@localhost hello-1.0]# ls
08. aclocal.m4 AUTHORS ChangeLog config.h.in configure configure.ac COPYING depcomp include 1
09. [root@localhost hello-1.0]#
10. [root@localhost hello-1.0]#
11. [root@localhost hello-1.0]# ./configure
12. checking for a BSD-compatible install... /usr/bin/install -c
13. checking whether build environment is sane... yes
14. checking for gawk... gawk
15. checking whether make sets $(MAKE)... yes
16. checking for gcc... gcc
17. checking for C compiler default output file name... a.out
18. checking whether the C compiler works... yes
19. checking whether we are cross compiling... no
20. checking for suffix of executables...
21. checking for suffix of object files... o
```

```
22. checking whether we are using the GNU C compiler... yes
23. checking whether gcc accepts -g... yes
24. checking for gcc option to accept ANSI C... none needed
25. checking for style of include used by make... GNU
26. checking dependency style of gcc... gcc3
27. configure: creating ./config.status
28. config.status: creating Makefile
29. config.status: creating config.h
30. config.status: executing depfiles commands
31. [root@localhost hello-1.0]#
32. [root@localhost hello-1.0]# make
33. make all-am
34. make[1]: Entering directory `/home/chenjie/hello-1.0'
35. if gcc -DHAVE_CONFIG_H -I. -I. -I. -g -O2 -MT main.o -MD -MP -MF ".deps/main.Tpo" -c -o main.o \
36. then mv -f ".deps/main.Tpo" ".deps/main.Po"; else rm -f ".deps/main.Tpo"; exit 1; fi
37. if gcc -DHAVE_CONFIG_H -I. -I. -I. -g -O2 -MT test.o -MD -MP -MF ".deps/test.Tpo" -c -o test.o \
38. then mv -f ".deps/test.Tpo" ".deps/test.Po"; else rm -f ".deps/test.Tpo"; exit 1; fi
39. gcc -g -O2 -o hello main.o test.o
40. make[1]: Leaving directory `/home/chenjie/hello-1.0'
41. [root@localhost hello-1.0]#
42. [root@localhost hello-1.0]# make install
43. make[1]: Entering directory `/home/chenjie/hello-1.0'
44. test -z "/usr/local/bin" || mkdir -p -- "/usr/local/bin"
45. /usr/bin/install -c 'hello' '/usr/local/bin/hello'
46. make[1]: Nothing to be done for `install-data-am'.
47. make[1]: Leaving directory `/home/chenjie/hello-1.0'
48. [root@localhost hello-1.0]#
49. [root@localhost hello-1.0]# hello
50. main entrance.
51. test method.
```

## 五、命令使用的整个流程图

图我就不画了，转载两个图[2][3]，对比着看，或许更明白一些。





## 六、总结

本文描述了如果使用GNU Autotools的来管理源代码，发布源代码包，以及获得源代码包后如何编译、安装。由于这个例子过于简单，GNU Autotools的用法还未完全描述清楚，主要体现在以下几点：

（1）在创建 Makefile.am 文件中，描述的很简单。在实际的项目中，文件关系很复杂，而且还有引用其他动态库、第三方动态库等关系。

（2）虽然 makefile 是自动生成的，但是了解它的规则是非常重要的。makefile 涉及到的规则本文并未加以描述。

有空的时候再写一篇blog来描述上述两个问题。

[1] <http://book.chinaitlab.com/linux/777286.html>

[2] <http://blog.ossxp.com/2010/04/954/>

顶 踩  
2 0

上一篇 Windows系统和Linux系统中的静态链接库与动态链接库(三)

下一篇 Source Insight的一点小技巧

## 我的同类文章

### UNIX ( 2 )

- Source Insight的一点小技巧 2010-12-18 阅读 4169
- linux下的FTP,SAMBA配置笔记 2008-10-21 阅读 3192



参考知识库



Go知识库

1804 关注 | 786 收录



Linux知识库

9335 关注 | 3511 收录

猜你在找

- Linux编程之GCC编译工具实战
- GNU Autotools的使用方法
- ArcGIS for javascript 项目实战（环境监测系统）
- AutoTools 疑难解决方法
- 从零开始学习mac系统视频录制编辑软件Camtasia For
- AutoTools 疑难解决方法
- 马哥运维教程-课前环境准备和上课软件环境介绍
- AutoTools 简单入门 转
- 2016高薪Linux培训教程-系统组成、文件系统及基础命
- Makefile使用详解

查看评论

3楼 [叶子还在书上](#) 2016-04-17 11:05发表



请问我可以转载到我的博客作为学习笔记吗？

2楼 [zhangwiwi](#) 2014-03-25 14:37发表



2.2 将configure.scan 文件重命名为configure.ac，并做适当的修改 楼主这边有个小错误,必须得将configure.scan 输入重命名为configure.in才行。

Re: [ic499747](#) 2016-02-29 10:13发表



回复zhangwiwi：重命名为configure.ac或configure.in都可以，用configure.ac是ok的，ubuntu环境

1楼 [on\\_1y](#) 2013-11-05 22:58发表



不错，很有帮助。

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题   Hadoop   AWS   移动游戏   Java   Android   iOS   Swift   智能硬件   Docker   OpenStack  
VPN   Spark   ERP   IE10   Eclipse   CRM   JavaScript   数据库   Ubuntu   NFC   WAP   jQuery  
BI   HTML5   Spring   Apache   .NET   API   HTML   SDK   IIS   Fedora   XML   LBS   Unity  
Splashtop   UML   components   Windows Mobile   Rails   QEMU   KDE   Cassandra   CloudStack   FTC  
coremail   OPhone   CouchBase   云计算   iOS6   Rackspace   Web App   SpringSide   Maemo  
Compuware   大数据   aptech   Perl   Tornado   Ruby   Hibernate   ThinkPHP   HBase   Pure   Solr  
Angular   Cloud Foundry   Redis   Scala   Django   Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服   杂志客服   微博客服   webmaster@csdn.net   400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

