Auto Tools基本使用

Auto Tools是基于GNU的自动为你的软件源代码生成Makefile以及编译并发布软件软件包等的一套软件包管理工具集合。这里，用一个具体的例子，来演示一下Autotools工具相对之前的"Auto Tools使用流程"例子(hello-1.0)比较"传统"的基本使用方法。假设程序名字为hello-2.0，通过这个例子，我们可以知道使用GNU Autotools 需要自己写哪些文件，如何编写这些文件，以及如何设置程序链接的静态库，如何指定库的安装与不安装，如何设置编译选项等等。

内容比较多，所以首先给出一个总体的使用方法，之后对每一个步骤进行详细的讲解。

主要内容：

一、总体步骤

二、详细解释

三、关于configure配置

四、文件详细信息

五、其它

一、总体步骤

==========

1,编写hello-2.0程序

查看最初代码目录结构如下：

#pwd

/root/test/hello-2.0

#ls

lib  src my\_doc

这里大致内容如下：

src/中的文件是程序的主文件。

lib/中的文件是程序需要链接的库。

lib/iface1中的库只是作为链接使用，并不安装在系统中。

lib/ifacex中的库不仅要链接，而且安装的时候要安装到系统中。

2,为程序hello-2.0添加Autotools模板文件

为了让Autotools可以自动生成符合标准GNU风格的Makefile，需要添加一些文件。实际上，我们必须写并且要向源代码中添加的文件就两种：configure.ac，Makefile.am。

3,生成配置文件

# autoreconf --install

运行这个命令之后会生成了许多其他的文件，其中核心的程序是configure脚本，通过它以及生成的Makefile.in文件，最终会生成需要的Makefile.

4,配置并生成Makefile

#mkdir hello\_build

#cd hello\_build

#../configure CONFIG\_SITE=$(pwd)/../config.site --prefix=$(pwd)/target

这里，会生成Makefile，生成Makefile之后，才能构进行后面的编译工作。

5,编译并安装软件

#make

#make install

运行make就是编译，运行make install会将生成的将要安装的文件复制到之前指定的prefix目录（这里就是target）中。

6,发布软件包

#make dist

这样就会对软件包进行发布。发布的内容被打包到一个tar包中，其中包含的内容没有编译中间文件，而是运行"autoreconf --install"命令之后的一些文件。

二、详细解释

==========

这里，对以上的步骤进一步解释，想要知道其中涉及到的文件的具体内容，可以参考后面的"文件详细信息"

1)编写hello-2.0程序

编写之后，使用find命令查看程序的源代码目录结构如下：

#cd hello-2.0

#find .

.

./my\_doc

./src

./src/main.c

./src/myprint.c

./src/myprint.h

./lib

./lib/iface1

./lib/iface1/print1.c

./lib/iface1/print1.h

./lib/ifacex

./lib/ifacex/print2.c

./lib/ifacex/print3.c

./lib/ifacex/printx.h

这里，为了方便阅读，我调整了输出的顺序。其中my\_doc文件是一个自己随便写的说明文件，记录本程序使用的方法供学习使用。根据输出，我们可以了解到，

src/中的文件是程序的主文件。

lib/中的文件是程序需要链接的库。

lib/iface1中的库只是作为链接使用，并不安装在系统中。

lib/ifacex中的库不仅要链接，而且安装的时候要安装到系统中。

2)为程序hello-2.0添加Autotools模板文件

添加之后，使用find命令查看源代码目录结构如下：

#cd hello-2.0

#find .

.

./my\_doc

./src

./src/main.c

./src/myprint.c

./src/myprint.h

./lib

./lib/iface1

./lib/iface1/print1.c

./lib/iface1/print1.h

./lib/ifacex

./lib/ifacex/print2.c

./lib/ifacex/print3.c

./lib/ifacex/printx.h

./configure.ac

./config.site

./Makefile.am

./src/Makefile.am

./lib/Makefile.am

./lib/iface1/Makefile.am

./lib/ifacex/Makefile.am

这里，为了方便阅读，我调整了输出顺序。根据输出我们可以知道，为了让Autotools可以自动生成符合标准GNU风格的Makefile，我们添加了不少的文件。下面分别对添加的文件的功能进行介绍。

a)./configure.ac

这个文件是Autoconf工具需要读取的模板文件，通过这个文件指定软件包一些全局的信息。

b)./config.site

这个文件是可的，是将要生成configure脚本的配置文件，configure可以通过这个配置文件设置一些选项。

c)./Makefile.am

./src/Makefile.am

./lib/Makefile.am

./lib/iface1/Makefile.am

./lib/ifacex/Makefile.am

这些Makefile.am文件都是是生成Makefile文件所需的模板文件，包含源代码文件以及一些编译选项信息。通过一个顶层的Makefile.am指定各个子Makefile.am位置，实际可以把所有文件内容集中到一个Makefile.am中，不过现在这样写，是为了了解关于Makefile.am更多的知识。

更多这些文件具体的信息，通过后面给出的文件详细内容可以了解。实际上，通过以上的介绍，我们可知，为了让Autotools真正开始工作，我们必须写并且要向源代码中添加的文件就两种：configure.ac，Makefile.am。

3)生成配置文件

# autoreconf --install

configure.ac: installing `build-aux/install-sh'

configure.ac: installing `build-aux/missing'

lib/iface1/Makefile.am: installing `build-aux/depcomp'

运行之后，查看代码目录如下：

#find .

.

./my\_doc

./src

./src/main.c

./src/myprint.c

./src/myprint.h

./lib

./lib/iface1

./lib/iface1/print1.c

./lib/iface1/print1.h

./lib/ifacex

./lib/ifacex/print2.c

./lib/ifacex/print3.c

./lib/ifacex/printx.h

./configure.ac

./config.site

./Makefile.am

./src/Makefile.am

./lib/Makefile.am

./lib/iface1/Makefile.am

./lib/ifacex/Makefile.am

./configure

./Makefile.in

./aclocal.m4

./config.h.in

./lib/Makefile.in

./lib/iface1/Makefile.in

./lib/ifacex/Makefile.in

./src/Makefile.in

./build-aux

./build-aux/missing

./build-aux/install-sh

./build-aux/depcomp

./autom4te.cache

./autom4te.cache/output.0

./autom4te.cache/requests

./autom4te.cache/traces.1

./autom4te.cache/output.1

./autom4te.cache/traces.0

这里，为了便于阅读，我调整了输出的顺序。可见，除了之前我们添加的文件之外，还生成了许多其他的文件，其中核心的程序是configure脚本，通过它以及生成的Makefile.in文件，最终会生成需要的Makefile.至此，这些文件也就是我们发布软件时候的文件了。下面我们看看编译的过程。

4)配置并生成Makefile

#mkdir hello\_build

#cd hello\_build

#../configure CONFIG\_SITE=$(pwd)/../config.site --prefix=$(pwd)/target

运行之后，查看代码目录如下：

# pwd

/root/test/hello-2.0/hello\_build

# find

.

./src

./src/.deps

./src/.deps/myprint.Po

./src/.deps/main.Po

./src/Makefile

./config.h

./lib

./lib/iface1

./lib/iface1/.deps

./lib/iface1/.deps/print1.Po

./lib/iface1/Makefile

./lib/ifacex

./lib/ifacex/.deps

./lib/ifacex/.deps/print2.Po

./lib/ifacex/.deps/print3.Po

./lib/ifacex/Makefile

./lib/Makefile

./config.log

./stamp-h1

./config.status

./Makefile

这里，生成Makefile之后，才能构进行后面的编译工作。

a)我们创建hello\_build目录的目的是想要将编译的中间文件集中到一个目录中而不会影响到原来的源代码的目录结构，因为configure程序的特点就是，在那里运行它，那么就会将哪里做为编译的初始目录;

b)我们使用CONFIG\_SITE环境变量设置configure读取的配置文件(config.site)的位置，这里要使用绝对路径，如果没有设置这个环境变量，那么就会寻找prefix/share/config.site文件，如果还没有就使用prefix/etc/config.site文件；

c)prefix就是使用--prefix指定的软件安装目录，后面做"make install"安装软件的时候，会将软件安装到这个目录下。

另外，如果我们在config.site指定用交叉编译(比如arm)，那么运行的命令类似如下 "#../configure CONFIG\_SITE=$(pwd)/../config.site --host=i686 --prefix=$(pwd)/target"这里使用--host指定编译主机。

5)编译并安装软件

#make

#make install

运行make就是为了编译软件，运行make install会将生成的将要安装的文件复制到之前指定的prefix目录（这里就是target）中。查看运行之后的目录结构如下：

# pwd

/root/test/hello-2.0/hello\_build

# find .

.

./src

./src/myprint.o

./src/.deps

./src/.deps/myprint.Po

./src/.deps/main.Po

./src/main.o

./src/hello

./src/Makefile

./config.h

./target

./target/bin

./target/bin/hello

./target/lib

./target/lib/libifacex.a

./target/include

./target/include/printx.h

./target/include/myprint.h

./lib

./lib/iface1

./lib/iface1/libiface1.a

./lib/iface1/print1.o

./lib/iface1/.deps

./lib/iface1/.deps/print1.Po

./lib/iface1/Makefile

./lib/ifacex

./lib/ifacex/print3.o

./lib/ifacex/libifacex.a

./lib/ifacex/print2.o

./lib/ifacex/.deps

./lib/ifacex/.deps/print2.Po

./lib/ifacex/.deps/print3.Po

./lib/ifacex/Makefile

./lib/Makefile

./config.log

./stamp-h1

./config.status

./Makefile

这里，我们就完成了软件的编译和安装。

6)发布软件包

#make dist

这样就会对软件包进行发布。查看这个命令之后的目录结构如下：

#ls -p

config.h  config.log  config.status  hello-2.0.tar.gz  lib/  Makefile  src/  stamp-h1  target/

从这里我们发现，实际就是生成了一个hello-2.0.tar.gz，也就是软件的发布包。查看其中的内容如下：

#tar -tzvf hello-2.0.tar.gz

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/

-rwxrwxrwx 500/500          20 2011-06-17 11:19:57 hello-2.0/Makefile.am

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/src/

-rwxr-xr-x 0/0             228 2011-06-17 11:25:40 hello-2.0/src/Makefile.am

-rwxrwxrwx 500/500         292 2011-06-17 13:47:46 hello-2.0/src/main.c

-rwxrwxrwx 500/500         141 2011-06-16 18:15:09 hello-2.0/src/myprint.c

-rwxrwxrwx 500/500          63 2011-01-31 15:32:46 hello-2.0/src/myprint.h

-rw-r--r-- 0/0           13581 2011-06-17 14:28:58 hello-2.0/src/Makefile.in

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/build-aux/

-rwxr-xr-x 0/0           11014 2011-06-17 14:28:58 hello-2.0/build-aux/missing

-rwxr-xr-x 0/0            9233 2011-06-17 14:28:58 hello-2.0/build-aux/install-sh

-rwxr-xr-x 0/0           15936 2011-06-17 14:28:58 hello-2.0/build-aux/depcomp

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/lib/

-rwxr-xr-x 0/0              24 2011-06-16 18:09:59 hello-2.0/lib/Makefile.am

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/lib/iface1/

-rwxr-xr-x 0/0             119 2011-06-16 18:13:32 hello-2.0/lib/iface1/Makefile.am

-rwxrwxrwx 500/500          78 2011-02-01 14:42:55 hello-2.0/lib/iface1/print1.c

-rwxrwxrwx 500/500          60 2010-10-19 11:58:30 hello-2.0/lib/iface1/print1.h

-rw-r--r-- 0/0           11388 2011-06-17 14:28:58 hello-2.0/lib/iface1/Makefile.in

-rw-r--r-- 0/0           13019 2011-06-17 14:28:58 hello-2.0/lib/Makefile.in

drwxr-xr-x 0/0               0 2011-06-17 14:49:23 hello-2.0/lib/ifacex/

-rwxrwxrwx 500/500          82 2011-06-16 18:08:28 hello-2.0/lib/ifacex/print2.c

-rwxr-xr-x 0/0             235 2011-06-17 10:34:32 hello-2.0/lib/ifacex/Makefile.am

-rwxr-xr-x 0/0              82 2011-06-16 18:08:39 hello-2.0/lib/ifacex/print3.c

-rwxrwxrwx 500/500          75 2011-06-16 18:07:56 hello-2.0/lib/ifacex/printx.h

-rw-r--r-- 0/0           13795 2011-06-17 14:28:58 hello-2.0/lib/ifacex/Makefile.in

-rw-r--r-- 0/0           31120 2011-06-17 14:28:56 hello-2.0/aclocal.m4

-rwxrwxrwx 500/500         878 2011-06-17 11:19:04 hello-2.0/configure.ac

-rwxr-xr-x 0/0          132936 2011-06-17 14:28:57 hello-2.0/configure

-rw-r--r-- 0/0           18068 2011-06-17 14:28:58 hello-2.0/Makefile.in

-rw-r--r-- 0/0             557 2011-06-17 14:28:57 hello-2.0/config.h.in

这里，实际我们发布软件包的时候，发布的文件一些是通过我们最初建立的那几个Makefile.am指定的，我们发布的文件都是运行"autoreconf --install"生成configure之后的、但是生成Makefile之前的那些文件。(注意，我们可以看到其中的config.site和自己建立的文档my\_doc并没有被包含进来。)

三、关于configure配置

==========

对于configure配置，生成的Makefile，以及编译安装之后的程序，我们需要了解如下一些知识。

1,一些常用的make目标

实际上，在前面我们还可以运行"make clean"，"make distclean"等命令，之前的讲解已经对这些做了实践，这里就不演示了。这里对一些常见的Makefile目标进行一点总结，如下：

"make all" 编译程序，库，以及文档等等（和"make"一样）。

"make install" 安装可执行程序。

"make install-strip" 和"make install"一样, 然后去掉调试信息。

"make uninstall" 和"make install"功能一样。

"make clean" 清除编译的中间文件（和"make all"功能相反）

"make distclean" 除了前面的"make clean"之外，还清除所有"./configure"创建的中间文件。

"make check" 如果有测试包的话，则运行之。

"make installcheck" 如果支持的化，就检查已经安装的程序或者库。

"make dist" 创建PACKAGE-VERSION.tar.gz发布包。

2,configure常用路径变量：

./configure和安装路径相关的常用变量：

变量名称 默认值

prefix "/usr/local"

exec-prefix prefix

bindir exec-prefix"/bin"

libdir exec-prefix"/lib"

...

includedir prefix"/include"

datarootdir prefix"/share"

datadir datarootdir

mandir datarootdir"/man"

infodir datarootdir"/info"

这些变量可以自行指定，例如指定prefix之后再进行安装,例如：

./configure --prefix ~/usr

make

make install

这里，--prefix后面可以有"="也可以没有。

3,configure常用配置选项变量：

./configure和配置相关的常用变量:

CC C编译器命令

CFLAGS C编译选项

CXX C++编译器命令

CXXFLAGS C++编译选项

LDFLAGS 链接选项

CPPFLAGS C/C++预处理选项

更多的项可以参见"./configure --help"

使用举例：

./configure --prefix ~/usr CC=gcc-3 CPPFLAGS=-I$HOME/usr/include LDFLAGS=-L$HOME/usr/lib

四、文件详细信息

==========

这里，列出了最初那些需要我们写的文件的内容,也注明了其中需要注意的内容。

src/main.c文件：

---------

#include <stdio.h>

#include "myprint.h"

int main(int argc, char \*argv[])

{

#ifdef MYDEBUG1

printf("Define MYDEBUG1\n");

#else

printf("Havn't define MYDEBUG1\n");

#endif

#ifdef MYDEBUG2

printf("Define MYDEBUG2\n");

#else

printf("Havn't define MYDEBUG2\n");

#endif

myprint();

return 0;

}

src/myprint.h文件：

---------

#ifndef \_\_MYPRINT\_H

#define \_\_MYPRINT\_H

void myprint();

#endif

src/myprint.c文件：

---------

#include "myprint.h"

#include "../lib/iface1/print1.h"

#include "../lib/ifacex/printx.h"

void myprint()

{

print1();

print2();

print3();

}

lib/iface1/print1.h

---------

#ifndef \_\_PRINT1\_H

#define \_\_PRINT1\_H

void print1();

#endif

lib/iface1/print1.c

---------

#include <stdio.h>

#include "print1.h"

void print1()

{

printf("print1\n");

}

lib/ifacex/printx.h

---------

#ifndef \_\_PRINTX\_H

#define \_\_PRINTX\_H

void print2();

void print3();

#endif

lib/ifacex/print2.c

---------

#include <stdio.h>

#include "printx.h"

void print2(void)

{

printf("print2\n");

}

lib/ifacex/print3.c

---------

#include <stdio.h>

#include "printx.h"

void print3(void)

{

printf("print3\n");

}

至此，列出的都是源代码，后面将列出Autotools需要的文件。

./configure.ac

---------

#This file is edited by QuietHeart.

#\*Package name, version, and bug report address.

AC\_INIT([hello], [2.0], [quiet\_heart000@126.com])

#A safe check make sure 'configure' is not run from outer space.

AC\_CONFIG\_SRCDIR([src/main.c])

#Auxiliary scripts such as install-sh and depcomp should be in this directory.

AC\_CONFIG\_AUX\_DIR([build-aux])

#Turn automake warnings and report them as errors. This is a foreign package.

AM\_INIT\_AUTOMAKE([-Wall -Werror foreign])

#Check for C compile.

AC\_PROG\_CC

#For libuse in Makefile.am

AC\_PROG\_RANLIB

#AC\_PROG\_LIBTOOL

#Declare config.h as output header.

AC\_CONFIG\_HEADERS([config.h])

#XXX Declare Makefile as output file

#!!!Note:The order should be the same as Makefile.am, indicate the build order.

AC\_CONFIG\_FILES([Makefile

lib/Makefile

lib/iface1/Makefile

lib/ifacex/Makefile

src/Makefile

])

#Output all declared files.

AC\_OUTPUT

这个文件，需要注意的一个地方就是使用AC\_CONFIG\_FILES指定的Makefile次序要和Makefile.am中指定的目录递归次序一直，并且这个次序也表明了编译的次序。

./config.site

---------

###### Compile tools.

#CC=arm-sony-linux-gnueabi-gcc

CC=gcc

#LD=arm-sony-linux-gnueabi-ld

#AR=arm-sony-linux-gnueabi-ar

#LDFLAGS+="-lsonypthread -lsonydl -lsonyrt -lsonyc -lsonygcc\_s "

###### Global options.

#!!!Note: You'd better put the variables value in quote,and add ' ' after param value.

CFLAGS+="-g "

CFLAGS+="-D MYDEBUG1 "

CFLAGS+="-D MYDEBUG2 "

###### Miscellaneous

#prefix=

#test -z "$CC" && CC=gcc-3

#test -z "$CPPFLAGS" && CPPFLAGS=-I$HOME/usr/include

#test -z "$LDFLAGS" && LDFLAGS=-L$HOME/usr/lib

这个文件需要注意的就是，编译选项的值要用""括起来，并且最后要有一个空格。

./Makefile.am

---------

SUBDIRS = . lib src

src/Makefile.am

---------

#AUTOMAKE\_OPTIONS=foreign

bin\_PROGRAMS=hello

hello\_SOURCES=

include\_HEADERS=

hello\_SOURCES+=main.c \

myprint.c

include\_HEADERS+=myprint.h

#!!!Note the path.

hello\_LDADD=../lib/iface1/libiface1.a ../lib/ifacex/libifacex.a

这里需要注意的是，使用hello\_LDADD可以指定hello需要链接的库，路径要使用相对当前Makefile.am的路径。如果需要专门指定hello这个程序的编译选项可以设置hello\_CFLAGS等，具体参见手册。

lib/Makefile.am

---------

SUBDIRS = iface1 ifacex

lib/iface1/Makefile.am

---------

#lib not installed only used for building.

noinst\_LIBRARIES=libiface1.a

libiface1\_a\_SOURCES= print1.c \

print1.h

lib/ifacex/Makefile.am

---------

#lib tobe installed

lib\_LIBRARIES=libifacex.a

#source code files.

#note: you can use 'find -name \*.c' if has too much source code.

libifacex\_a\_SOURCES= print2.c \

print3.c

#header files to be installed.

include\_HEADERS=printx.h

这里个技巧，就是如果代码文件太多了的话Makefile.am也会很多，我们可以用类似"find -name \*.c"的命令来输出相应的文件，用"+="的赋值的方式统一处理添加。

五、其它

==========

参考资料：

http://www.gnu.org/software/autoconf/manual/autoconf.html#Site-Configuration

本文只是在实践的基础上做了一个基本的使用介绍，记录下来，便于查阅。关于这个工具还有更多的功能，如有机会，随着学习和应用实践得增多，会继续增加这一部分的相应内容，希望本文能够给需要的读者带来帮助，如果有问题或者更好的建议，可以联系我，互相学习。^\_^

作者：QuietHeart

Email：quiet\_heart000@126.com

2011年6月17日。