Autotool--configure 参数说明

作者：ReyRen
原文链接：https://ld246.com/article/1564659291905
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20171120.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h3 id="configuring">configuring</h3>
<code>configure</code> 脚本使用了大量的命令行选项. 可以通过 <code>configure --help
/code> 查看所支持的选项. 这些选项很多但是好多都晦涩难懂. 所以说如果想要定制化的安装, 还是
要了解选项的意义的. 这里罗列出的是一些简短的解释:
--cache-file=file

我们知道 <code>configure</code> 的作用就是检查当前系统对一些功能和函数等的支持(bug!!).
么每次执行后的结果文件可以存在一个缓存文件中, 这样有利于之后的一系列 <code>configure</co
e> 的调用. 在一些较为大型的项目中, 拥有者好多的源码树. 在每一个源码树上都有自己 <code>confi
ure</code>.
--help

顾名思义, 打印出帮助信息. 及时是经验丰富人士, 偶尔也还是需要这个选项来开一下下的. 比如 <code
configure</code> 在 GCC 的包中就允许你掌控石否需要编译 GNU 的汇编器还是第三方的汇编器.<
r>
--create

<code>configure</code> 文件的主要一个作用是生成执行后的文件, 那么这个选项就是让不要导
生成的文件. 你可以理解为"干运行". 但是缓存中是修改了的.
--quiet
--silent

<code>configure</code> 文件在运行的时候会输出一些信息来告诉用户在执行的脚本是什么, 执
的情况怎么样等. 这样做是因为 <code>configure</code> 可能会执行的很慢. 不要让用户过于焦虑,
所以默认会输出一些细节. 但是加上这个参数后, 就是让用户焦虑的等下去的节奏.
--version

打印出生成 <code>configure</code> 脚本的 Autoconf 的版本.
--prefix=prefix

这个是最常见到的选项之一. 如果在生成 Makefile 的时候传入了这个参数, 那么很有可能会在安装的
候重新定位系统的独立性. 简单的说就是一些文件或者软件安装位置不同了. 比如说安装 Emacs:
<pre><code class="language-bash highlight-chroma"><span c
ass="highlight-cl">$./configure --prefix=/opt/gnu
</code></pre>
这个语句将会使得 Emacs 的 List 文件安装到 <code>/opt/gnu/share</code> 下.
这个是相当重要的, 一定得利用生成的文件合理的利用这个条件. 在之后会介绍 <code>Automake</
ode> 将会简化这个过程. 开发人员只需要创建这些文件即可.
--exec-prefix=eprefix

首先说一下频繁出现的一个名词 <code>architecture-dependent</code>. 这个词直白的翻译过来
是体系结结构相关的. 然后在解释这个选项参数, 这个参数作用类似于 prefix, 但是不同点是这个参数
按照体系结构相关性将文件安装到相关路径.
--bindir=dir

指定二进制文件安装路径. 当然会一个程序会生成很多的二进制文件, 这里指定的二进制是用户可以直使用的二进制.
--sbindir=dir

这个是指定管理员二进制文件的位置. 通常来讲这些二进制只能 root 用户进行操作.
--libexecdir=dir

可执行文件的存放位置, 但是这里的可执行文件通常意义并不是用户可以直接使用的. 类比于之前的 biary 那个选项. 这里的可执行文件比如 linux 支持的 ELF 等. 这里的可执行文件好多是由 binary 指定文件调用的.
--datadir=dir

普通文件的指定路径.
--sysconfdir=dir

用于单机操作的只读文件的存放路径.
--sharedstatedir=dir

这里指定的是一些可以进行修改和可以对多主机共享的文件路径.
--localstatedir=dir

这里指定的是一些可以进行修改和只能对本级有效的文件的路径.
--libdir=dir

目标代码库的指定路径.
--includedir=dir

C 头文件的指定路径. 当然像其他语言的头文件比如 C++ 头文件也是存放在这里的.
--oldincludedir=dir

这里指定的是除了对 GCC 这个编译器用的 C 头文件外其他编译器用到的头文件的指定路径.
--infodir=dir

指定 Info 格式的文件安装路径. Info 格式是 GNU 项目定一种文件格式.
--mandir=dir

mans 手册安装路径.
--srcdir=dir

这个选项并不会影响编译, 但是它是告诉 <code>configure</code> 源码在哪. 这个选项通常是不需指定的. 因为 <code>configure</code> 和源文件一般是存放在一个目录下的.

--program-prefix=prefix

指定编译完的程序的前缀名字. 比如说, 又一个程序编译完后是叫做"tar", 如果用 prefix 是"g", 那么最
的程序名字将会是"gtar". 需要注意的是这个选项只有在"Makefile.in"中使用时才会起作用.
--program-suffix=suffix

这个毋庸置疑就是程序名字的后缀了.
--program-transform-name=program

这里的 program 是一个 sed 脚本文件, 当程序安装后, 这个程序的名字就会由 <code>sed -s progr
m</code> 来产生出来.
--build=build

指定包编译的时候的系统类型. 如果没有指定的话, 就是和下面的"--host"指定的一样.
--host=host

这个是指定包将要运行的系统类型, 如果没有指定, 这个 host 会根据执行 <code>config.guess</cod
> 来抉择.
--target=target

这个是指定包指向的系统类型, 理解起来比较抽象哈, 所以这个选项一般用在变成语言的编译器或者汇
器的上下文中. 同样的, 如果没有没有指定, 同上面的"--host".
--disable-feature

在有些项目中是提供编译时可配置的功能的, 这样做也是在大规模的项目中能够在编译过程中对一些 fe
ture 的打开与关闭. 比如说 Kerberos 认证系统. 如果默认情况下某个 feature 是打开的, 那么就可以
这个配置选项讲其关掉, "feature"就是 feature 名字: <code>$ ./configure --disable-gui </code>
br>
--enable-feature [=arg]

这个和上面的正好相反. 但是有的 feature 还能指定参数: <code>$ ./configure --enable-buffers=1
8 </code> 如果将=no, 那么就和上面的 disable 是一样的效果了.
--with-package [=arg]

在开源社区中包的共享使用和复用是极其常见的事情. 所以会有一些依赖的存在. 在使用'configure'进
配置的时候, 需要给定一些暗示已经安装的依赖包在哪里: BLT 工具包依赖于 Tcl 和 Tk 包. 为了配置 BL
:</p>
<pre><code class="language-bash highlight-chroma"><span c
ass="highlight-cl">$ ./configure --with-tcl=/usr/local --w
th-tk=/usr/local
</code></pre>
<p>用 <code>--with-package=no</code> 和下面的 <code>--without-package</code> 是
样的效果.
--without-package

原文链接：Autotool--configure 参数说明

有时候你是不希望你的包与系统上已经安装的包进行交互的. 比如说配置新的编译器的包, 你不希望使
GNU ld 进行链接:
<pre><code class="language-bash highlight-chroma"><span c
ass="highlight-cl">$ ./configure --without-gnu-ld
</code></pre>
--x-includes=dir

这个选项是"--with-package"的一个特殊情况. 在 Autoconf 发展之初, 用'configure'在 X Window
统上作为 Imake 的替代编译程序是很普遍的, 所以这个选项就是指引 configure 脚本去有 X11 头文
的目录去干活.
--x-libraries=dir

同理, 指引 configure 文件去有 X11 库的目录干活.
<hr>
在源码树中执行 <code>configure</code> 脚本来进行配置检测是没有必要也是极其不推荐的.
相反的, <code>configure</code> 文件生成的 Makefile 去跳到源码树中编译源码, 是大大的有好
的. 好处就是 <code>configure</code> 生成的副产物会让源码树中变的杂乱无章. 这样会使得同样
源码在不同的 <code>configure</code> 下编译变得异常艰难. 相反, 在编译过程中是推荐构建三种
: 源码树, 编译树和安装树. 这里给出一个编译 GNU 的 malloc 包的例子:
<pre><code class="highlight-chroma"><span class="highlight
cl">$ gtar zxf mmalloc-1.0.tar.gz
$ mkdir build &a
p;& cd build
$ ../mmalloc-1.0/
onfigure
creating cache ./c
nfig.cache
checking for gcc...
gcc
checking whether
he C compiler (gcc) works... yes
checking whether
he C compiler (gcc) is a cross-compiler... no
checking whether
we are using GNU C... yes
checking whether
gcc accepts -g... yes
checking for a BS
compatible install... /usr/bin/install -c
checking host sys
em type... i586-pc-linux-gnu
checking build sys
em type... i586-pc-linux-gnu
checking for ar... a

checking for ranlib
.. ranlib
checking how to r
n the C preprocessor... gcc -E
checking for unist
.h... yes

checking for getpgesize... yes
checking for working mmap... yes
checking for limits.h... yes
checking for stddef.h... yes
updating cache ../config.cache
creating ./config.status
</code></pre>
现在这个编译树就已经配置结束了. 接下来就是执行编译和安装到'/usr/local'下了:
<pre><code class="language-bash highlight-chroma">$ make all & make install
</code></pre>
<hr>
好了, 这一节就到这里, 接下来的一节将简单的介绍 <code>configure</code> 的产物, Makefile target(常用的), 在 Autoconf 中的"configuration name".