[./configure 参数解析](http://blog.csdn.net/caijp1090/article/details/7422460)

  Linux环境下的软件安装，并不是一件容易的事情；如果通过源代码编译后再安装，当然事情就更为复杂一些；现在安装各种软件的教程都非常普遍；但万变不离其宗，对基础知识的扎实掌握，安装各种软件的问题就迎刃而解了。Configure脚本配置工具就是基础之一，它是autoconf的工具的基本应用。  
'configure'脚本有大量的命令行选项。对不同的软件包来说，这些选项可能会有变化，但是许多基本的选项是不会改变的。带上'--help'选项执行'configure'脚本可以看到可用的所有选项。尽管许多选项是很少用到的，但是当你为了特殊的需求而configure一个包时，知道他们的存在是很有益处的。下面对每一个选项进行简略的介绍：

--cache-file=FILE

'configure'会在你的系统上[**测试**](http://lib.csdn.net/base/softwaretest)存在的特性(或者bug!)。为了加速随后进行的配置，测试的结果会存储在一个cache file里。当configure一个每个子树里都有'configure'脚本的复杂的源码树时，一个很好的cache file的存在会有很大帮助。

--help

输出帮助信息。即使是有经验的用户也偶尔需要使用'--help'选项，因为一个复杂的项目会包含附加的选项。例如，GCC包里的'configure'脚本就包含了允许你控制是否生成和在GCC中使用GNU汇编器的选项。

--no-create

'configure'中的一个主要函数会制作输出文件。此选项阻止'configure'生成这个文件。你可以认为这是一种演习(dry run)，尽管缓存(cache)仍然被改写了。

--quiet

--silent

当'configure'进行他的测试时，会输出简要的信息来告诉用户正在作什么。这样作是因为'configure'可能会比较慢，没有这种输出的话用户将会被扔在一旁疑惑正在发生什么，使用这两个选项中的任何一个都会把你扔到一旁。(译注：这两句话比较有意思，原文是这样的：If there was no such output, the user would be left wondering what is happening. By using this option, you too can be left wondering!)

--version

打印用来产生'configure'脚本的Autoconf的版本号。

--prefix=PEWFIX

'--prefix'是最常用的选项。制作出的'Makefile'会查看随此选项传递的参数，当一个包在安装时可以彻底的重新安置他的结构独立部分。举一个例子，当安装一个包，例如说Emacs，下面的命令将会使Emacs Lisp file被安装到"/opt/gnu/share"：

$ ./configure --prefix=/opt/gnu

--exec-prefix=EPREFIX

与'--prefix'选项类似，但是他是用来设置结构倚赖的文件的安装位置，编译好的'emacs'二进制文件就是这样一个文件。如果没有设置这个选项的话，默认使用的选项值将被设为和'--prefix'选项值一样。

--bindir=DIR

指定二进制文件的安装位置，这里的二进制文件定义为可以被用户直接执行的程序。

--sbindir=DIR

指定超级二进制文件的安装位置。这是一些通常只能由超级用户执行的程序。

--libexecdir=DIR

指定可执行支持文件的安装位置。与二进制文件相反，这些文件从来不直接由用户执行，但是可以被上面提到的二进制文件所执行。

--datadir=DIR

指定通用数据文件的安装位置。

--sysconfdir=DIR

指定在单个机器上使用的只读数据的安装位置。

--sharedstatedir=DIR

指定可以在多个机器上共享的可写数据的安装位置。

--localstatedir=DIR

指定只能单机使用的可写数据的安装位置。

--libdir=DIR

指定库文件的安装位置。

--includedir=DIR

指定C头文件的安装位置。其他语言如C++的头文件也可以使用此选项。

--oldincludedir=DIR

指定为除GCC外编译器安装的C头文件的安装位置。

--infodir=DIR

指定Info格式文档的安装位置.Info是被GNU工程所使用的文档格式。

--mandir=DIR

指定手册页的安装位置。

--srcdir=DIR

这个选项对安装没有作用，他会告诉'configure'源码的位置。一般来说不用指定此选项，因为'configure'脚本一般和源码文件在同一个目录下。

--program-prefix=PREFIX

指定将被加到所安装程序的名字上的前缀。例如，使用'--program-prefix=g'来configure一个名为'tar'的程序将会使安装的程序被命名为'gtar'。当和其他的安装选项一起使用时，这个选项只有当他被`Makefile.in'文件使用时才会工作。

--program-suffix=SUFFIX

指定将被加到所安装程序的名字上的后缀。

--program-transform-name=PROGRAM

这里的PROGRAM是一个sed脚本。当一个程序被安装时，他的名字将经过`sed -e PROGRAM'来产生安装的名字。

--build=BUILD

指定软件包安装的系统平台。如果没有指定，默认值将是'--host'选项的值。

--host=HOST

指定软件运行的系统平台。如果没有指定。将会运行`config.guess'来检测。

--target=GARGET

指定软件面向(target to)的系统平台。这主要在程序语言工具如编译器和汇编器上下文中起作用。如果没有指定，默认将使用'--host'选项的值。

--disable-FEATURE

一些软件包可以选择这个选项来提供为大型选项的编译时配置，例如使用Kerberos认证系统或者一个实验性的编译器最优配置。如果默认是提供这些特性，可以使用'--disable-FEATURE'来禁用它，这里'FEATURE'是特性的名字，例如：

$ ./configure --disable-gui

-enable-FEATURE[=ARG]

相反的，一些软件包可能提供了一些默认被禁止的特性,可以使用'--enable-FEATURE'来起用它。这里'FEATURE'是特性的名字。一个特性可能会接受一个可选的参数。例如：

$ ./configure --enable-buffers=128

`--enable-FEATURE=no'与上面提到的'--disable-FEATURE'是同义的。

--with-PACKAGE[=ARG]

在自由软件社区里，有使用已有软件包和库的优秀传统。当用'configure'来配置一个源码树时，可以提供其他已经安装的软件包的信息。例如，倚赖于Tcl和Tk的BLT器件工具包。要配置BLT，可能需要给'configure'提供一些关于我们把Tcl和Tk装的何处的信息：

$ ./configure --with-tcl=/usr/local --with-tk=/usr/local

'--with-PACKAGE=no'与下面将提到的'--without-PACKAGE'是同义的。

--without-PACKAGE

有时候你可能不想让你的软件包与系统已有的软件包交互。例如，你可能不想让你的新编译器使用GNU ld。通过使用这个选项可以做到这一点：

$ ./configure --without-gnu-ld

--x-includes=DIR

这个选项是'--with-PACKAGE'选项的一个特例。在Autoconf最初被开发出来时，流行使用'configure'来作为Imake的一个变通方法来制作运行于X的软件。'--x-includes'选项提供了向'configure'脚本指明包含X11头文件的目录的方法。

--x-libraries=DIR

类似的，'--x-libraries'选项提供了向'configure'脚本指明包含X11库的目录的方法。

在源码树中运行'configure'是不必要的同时也是不好的。一个由'configure'产生的良好的'Makefile'可以构筑源码属于另一棵树的软件包。在一个独立于源码的树中构筑派生的文件的好处是很明显的：派生的文件，如目标文件，会凌乱的散布于源码树。这也使在另一个不同的系统或用不同的配置选项构筑同样的目标文件非常困难。建议使用三棵树：一棵源码树(source tree)，一棵构筑树(build tree)，一棵安装树(install tree)。这里有一个很接近的例子，是使用这种方法来构筑GNU malloc包：

$ gtar zxf mmalloc-1.0.tar.gz

$ mkdir build && cd build

$ ../mmalloc-1.0/configure

creating cache ./config.cache

checking for gcc... gcc

checking whether the C compiler (gcc ) works... yes

checking whether the 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 BSD compatible install... /usr/bin/install -c

checking host system type... i586-pc-[**Linux**](http://lib.csdn.net/base/linux)-gnu

checking build system type... i586-pc-[**linux**](http://lib.csdn.net/base/linux)-gnu

checking for ar... ar

checking for ranlib... ranlib

checking how to run the C preprocessor... gcc -E

checking for unistd.h... yes

checking for getpagesize... yes

checking for working mmap... yes

checking for limits.h... yes

checking for stddef.h... yes

updating cache ../config.cache

creating ./config.status

这样这棵构筑树就被配置了，下面可以继续构筑和安装这个包到默认的位置'/usr/local'：

$ make all && make install

./configure 参数详解   

--prefix=PREFIX

把所有文件装在目录   PREFIX下面而不是   /usr/local/pgsql   里．实际的文件会安装到不同的子目录里；甚至没有一个文件会直接   安装到   PREFIX   目录里．       
      
如果你有特殊需要，你还可以用下面的选项客户化不同的子目录的位置．       
      
--exec-prefix=EXEC-PREFIX     
你可以把体系相关的文件安装到一个不同的位置，   EXEC-PREFIX，而不是   PREFIX   设置的地方．这样做可以比较方便地在不同主机之间共享体系相关的文件。如果你省略这些，那么   EXEC-PREFIX   就会被设置为等于   PREFIX   并且体系相关和体系无关的文件都会安装到同一棵目录树下，   这也可能是你想要的．       
      
--bindir=DIRECTORY     
为可执行程序声明目录，缺省是   EXEC-PREFIX/bin，   通常也就是   /usr/local/pgsql/bin．       
      
--datadir=DIRECTORY     
设置所安装的程序需要的只读文件的目录．缺省是   PREFIX/share．   请注意这个目录和你放数据库文件的地方没有任何关系．       
      
--sysconfdir=DIRECTORY     
用于各种各样配置文件的目录，缺省为   PREFIX/etc．       
      
--libdir=DIRECTORY     
库文件和动态装载模块的目录．缺省是   EXEC-PREFIX/lib．       
      
--includedir=DIRECTORY     
C   和   C++   头文件的目录．缺省是   PREFIX/include．       
      
--docdir=DIRECTORY     
文档文件，（除   “man（手册页）”以外，   将被安装到这个目录．缺省是   PREFIX/doc．       
      
--mandir=DIRECTORY     
随着   PostgreSQL   一起带的手册页   将安装到这个目录．在它们相应的   manx子目录里．   缺省是   PREFIX/man．       
      
注意:   为了减少对共享安装位置（比如   /usr/local/include）   的污染，configure   自动在   datadir，   sysconfdir，includedir，   和   docdir   上附加一个   “/postgresql”   字串，   除非完全展开以后的目录名字已经包含字串   “postgres”   或者   “pgsql”．比如，如果你选择   /usr/local   做前缀，那么   C   的头文件将安装到   /usr/local/include/postgresql，   但是如果前缀是   /opt/postgres，那么它们将   被放进   /opt/postgres/include．       
      
      
      
      
--with-includes=DIRECTORIES     
DIRECTORIES   是一系列冒号分隔的目录，这些目录将被加入编译器的头文件   搜索列表中．如果你有一些可选的包（比如   GNU   Readline）安装在   非标准位置，你就必须使用这个选项，以及可能还有相应的   --with-libraries   选项．       
      
例子：--with-includes=/opt/gnu/include:/usr/sup/include．       
      
--with-libraries=DIRECTORIES     
DIRECTORIES   是一系列冒号分隔的目录，这些目录是用于查找库文件的．   如果你有一些包安装在非标准位置，你可能就需要使用这个选项   （以及对应的--with-includes选项）．       
      
例子：--with-libraries=/opt/gnu/lib:/usr/sup/lib．       
      
--enable-locale     
打开区域支持．区域支持会带来性能损失，但是如果   你不是在说英语的环境使用数据库，那么你很可能需要这个选项．       
      
--enable-recode     
打开单字节字符集记录的支持．参阅   Section   5.3   获取关于这个特性的详细信息．       
      
--enable-multibyte     
允许使用多字节编码．这个选项主要是用于象日语，韩语和中文   这样的语言．阅读   Section   5.2   获取详细信息．       
      
--with-pgport=NUMBER     
把   NUMBER   设置为服务器和客户端的缺省端口．缺省是   5432．   这个端口可以在以后设置，不过如果你在这里声明，那么   服务器和客户端将有相同的编译好了的缺省值．这样会方便些．       
      
--with-CXX     
制作C++支持库．       
      
--with-perl     
制作   Perl   接口模块．Perl   接口将被安装到   Perl   模块通常的   安装位置（典型的位置是   /usr/lib/perl），   所以你必须要有   root   权限来执行安装步骤（参阅   step   4）．使用这个选项你需要安装   Perl   5．       
      
--with-python     
制作   Python   接口模块．你需要   root   权限将   Python   模块安装到它的   缺省位置   （/usr/lib/pythonx.y）．   要使用这个选项，你必须已经安装了   Python   并且你的系统必须支持   共享库．如果你想制作一个新的完整的二进制解释器，你只能自己   手工制作．       
      
--with-tcl     
制作那些需要   Tcl/Tk   的部件，   就是   libpgtcl，pgtclsh，pgtksh，pgaccess和PL/Tcl．   你也可以看看下面的   --without-tk．       
      
--without-tk     
如果你同时声明了   --with-tcl   和这个选项，那么那些   需要   Tk（也就是   pgtksh   和pgaccess）的程序将被排除在外．       
      
--with-tclconfig=DIRECTORY,   --with-tkconfig=DIRECTORY     
Tcl/Tk   的安装文件   tclConfig.sh   和   tkConfig.sh，   她们里面包含制作   Tcl   或   Tk   模块接口的配置信息．   Configure   通常会自动在她们的众所周知的位置找到这些文件，但是如果你   需要一个不同版本的   Tcl   或   Tk，你也可以声明可以找到她们的目录．       
      
--enable-odbc     
制作   ODBC   驱动包．       
      
--with-odbcinst=DIRECTORY     
声明   ODBC   期望的odbcinst.ini配置文件的路径．   缺省是   /usr/local/pgsql/etc   或者你声明的   --sysconfdir   选项的内容．   安装程序将会把一个缺省的文件安装到那里．       
      
--with-krb4=DIRECTORY,   --with-krb5=DIRECTORY     
制作支持Kerberos认证的东西．你可以选择   Kerberos   版本   4   或   5，但不能同时选两个．   DIRECTORY   参数声明   Kerberos   安装的根目录；缺省假设是   /usr/athena   如果相关的头文件和库文件不在常用的父目录，那么你必须   使用附加的   --with-includes   和   --with-libraries   选项．另一方面，如果所需要的文件位于一个缺省的搜索目录，   （比如说，/usr/lib），   那你就不用输入这些参数了．       
      
configure   将在安装之前检查所需要的头文件和库文件以确信你的   Kerberos安装是完整的．       
      
--with-krb-srvnam=NAME     
Kerberos服务主的名称．   缺省是   “postgres”．通常没有理由改变这个值．       
      
--with-openssl=DIRECTORY     
制作支持   SSL   （加密的）联接的postgres．   这个选项需要安装   OpenSSL   包．   DIRECTORY   参数声明   OpenSSL   安装的根目录；缺省时   /usr/local/ssl．       
      
configure   将在安装之前检查所需要的头文件和库文件以确信你的   OpenSSL   安装是充分的．       
      
--with-java     
制作   JDBC   驱动以及相关的   Java   包．   这个选项要求你先安装   Ant   （当然还要有   JDK）．   请参考程序员手册   里面   JDBC   驱动的文档获取更多信息．       
      
--enable-syslog     
打开PostgreSQL   服务器使用   syslog   日志系统的功能．   （使用这个功能并不意味着你必须用   syslog   做日志，也不是说   服务器缺省会做这些，而是给你一个在运行时使用这个选项目的可能．）       
      
--enable-debug     
把所有程序和库以带有调试符号的方式编译．   这意味着你可以通过一个调试器运行程序来分析问题．   这样做显著增大了最后安装的可执行文件的大小，   并且在非   gcc   的编译器上，这么做通常还要关闭编译器优化，   导致速度的下降．但是，如果有这些符号表的话，就可以极大   帮助定位可能发生问题的位置．目前，我们认为这个选项对于   生产用途而言是边际变量，但是如果你正在进行开发工作，或者正在使用   beta   版本，   那么你就应该打开它．       
      
--enable-cassert     
打开在服务器中的   assertion   检查，   它会检查许多“不可能发生”的条件．它对于代码开发的用途   而言是无价之宝，不过这些测试稍微地减慢了一些速度．   这些断言检查并不一定都是针对严重错误的，因此一些相对无害的   小虫子也可能导致   postmaster   重启--只要它触发了一次断言失败．   目前，我们不推荐在生产环境中使用这个选项，但是如果你在做开发   或者在使用   beta   版本的时候应该打开它．       
      
      
如果你喜欢用那些和   configure   找出来的不同的   编译器，那么你可以分别设置你的环境变量   CC   和   CXX，把它们设置为你选择的程序．   类似地，你可以用   CFLAGS   和   CXXFLAGS   变量覆盖缺省编译器标志．比如：       
      
env   CC=/opt/bin/gcc   CFLAGS='-02   -pipe'   ./configure