# 第二章 VTK安装

这一章介绍在你的计算机系统中安装VTK所需要的步骤。这个过程的难点依赖若干因素。在Windows系统中，你可以安装预编译文件vtk.exe，然后可以用它来运行Tcl脚本文件。对于Python和Java的使用，可以链接VTK库到你自己的应用程序中去，或者在其他平台下使用VTK，而不一定是Windows系统，但这些必须通过VTK源码来编译VTK(因为存在有各种各样的平台，要保证各种平台下VTK都能正常运行，需要做的工作非常多，所以我们力求让VTK在各种平台下的编译尽可能的简单)。如果你选择的是从源码编译VTK，建立自己的函数库，在多核处理器系统中，可能会花费半小时左右或者更快；如果对于内存有限的系统，可能就要花费几个小时甚至更长的时间。另外，编译VTK所需时间的长短也取决于你要封装多少种解释性语言到VTK的C++内核中去以及你的系统配置。

你可以参考第三章“VTK系统架构”部分，这部分内容是关于VTK系统架构的概览，可能会使你接下来的编译过程更加顺得、轻松。如果在这个过程中遇到什么问题，你可以联系 vtkusers邮件列表（参考第一章“其他资源”一节）。

## 2.1 概览

VTK可以在不同的计算机平台下编译和运行。对于不同的平台，我们必须要考虑不同的操作系统、硬件配置以及编译器等的组合，由于这种组合的可能性比较多，所以发布VTK的二进制版本通常不大合适。因此，建议通过VTK源码来安装VTK，通过编译源码来生成VTK库文件和可执行程序（使用Tcl解释性语言创建VTK应用程序除外）。这样，预编译的二进制文件就可以是基于特定的系统平台。否则，你必须从VTK源码中编译Tcl以及Python的VTK可执行文件。

本章首先对CMake作一个总体介绍，CMake是一个跨平台的工程构建工具，可用于所有的操作系统配置编译环境。接着，本章内容按不同类型的操作系统分为两个部分，这样你就可以在Windows或者Unix平台下安装VTK。在安装过程中最好查看一些相关文档，VTK不支持早期版本的Windows，比如Windows 3.1等，同样也不支持版本老于OSX 10.2 (Jaguar)的Macintosh操作系统。

## 2.2 CMake

CMake是一个开源的、跨平台的用于配置、管理工程构建过程的工具。简单地说，平台独立文件(CMakeLists.txt)描述了工程构建的一些细节以及所要依赖的库文件。运行CMake时，它会根据你运行的特定的操作系统以及编译器创建本地编译文件。比如说，在Windows平台下，使用Microsoft Visual Studio编译器，项目(Solution)文件以及工程(Project)文件就会自动生成。而在Unix平台下，则是创建了makefile文件。通过这种方式，你就可以比较容易地在任何计算机平台下编译VTK，以及使用开发工具(如Editors, Debuggers, Profiles, Compilers等)(了解更多关于CMake的内容可以访问<http://www.cmake.org>，或者参考Kitware出版的《Mastering CMake》一书)。你可以从CMake网站上下载最新的CMake版本。

运行CMake时，需要提供三方面的信息：使用什么编译器；源文件目录的路径和编译目录的路径。编译目录是指编译过程中产生的目标代码、库文件以及二进制文件等存放的位置。CMake会从源文件目录中读取最顶层的CMakeLists.txt文件，然后在编译目录中生成缓存文件CMakeCache.txt。注意，CMake可以处理复杂的源文件目录结构，但要求源文件目录树下的每个子目录里必须要有一个CMakeLists.txt文件。

一旦这些基本的信息都提供了，用户就可以开始“Configure”。这一步里CMake会读取最顶层的CMakeLists.txt文件，以确定系统配置，定位系统资源以及遍历源文件目录中的每一层子目录。CMake运行时，会出现很多控制工程构建过程的CMake变量或者标志(称为CMake的缓存条目(CMake Cache Entries))。“Configure”完成以后，这些条目就会显示在CMake的GUI界面上，条目的显示风格可以通过CMake GUI上的视图选项进行切换。当改变某个条目的值时，要重新进行“Configure”。直到所有的CMake条目都配置完成以后，就可以进入下一步，即“Generate”。“Generate”阶段会根据特定的编译器生成项目文件、工程文件或者Makefiles文件。

接下来的两部分内容(Windows和Unix)详细介绍了在各个平台下CMake的运行步骤。注意，上面提到的步骤对所有的平台都是适用的。CMake界面可能会依据不同的操作系统有所不同。但是CMake-gui从CMake 2.6开始及后续版本都是基于Qt的，所以在不同的平台下CMake界面都是类似的。建议选择安装CMake的预编译二进制文件而不是从CMake源文件中安装，除非你想要研究CMake的源代码。

## 2.3 Windows XP, Vista及以上版本平台下安装VTK

在Windows平台下有两种安装VTK的方式。第一种是二进制/可执行文件安装，通过这种安装方式可以捆绑安装Tcl等工具（注：VTK的二进制安装文件可以从https://bitbucket.org/shuiling/vislab上下载到，这个文件里的VTK版本是5.0的，比较老，所以不建议用这种傻瓜式的安装方法安装VTK，同时建议安装下面内容提到的vtk-5.6.1-win32.exe文件，安装这个文件以后你可以在自己的计算机里运行\*.tcl的文件）。第二种方式是通过编译VTK源文件安装VTK，生成C++库文件以及VTK封装代码(用于生成Java, Tcl和Python等的可执行文件)。两种安装方式相比较而言，二进制安装方式要简单得多，但编译源文件安装VTK的方式则更具优势，你可以跟踪、调试以及更改VTK的源代码，如果你是开发人员的话，这种安装方式将会非常适合你。当然，如果你选择二进制安装VTK的话，你同样可以通过不同的方式扩展VTK：创建你自己的类(参考“编写VTK的类：概览”一节)；使用运行时可编程过滤器(Run-time Programmable Filters)(见“可编程过滤器”一节)以及运行时使用自己的VTK类取代VTK原有的类(见“对象工厂”一节)同样可以扩展VTK。

**二进制安装**

安装vtk.exe(用于运行Tcl/Tk文件的可执行程序)。可从VTK官网中下载到，一般是vtk-X.X.X-win32.exe，如vtk-5.6.1-win32.exe。图2-1所示，双击该文件以后就进入安装过程。安装文件名中的”X.X.X”表示即将安装的VTK的版本号。同时还可以下载相应版本号的VTK源代码的文件\*.zip以及VTK数据文件VTKData。每当发布新版本的VTK时，我们会在VTK的官网上给出链接，以供下载，用户可以关注vtkusers邮件列表上有关VTK版本发布的公告。

VTK源文件目录中包含了很多\*.tcl的脚本文件，借助这些文件你可以了解一下VTK里的类是如何协同工作的。把下载到的vtk-X.X.X.zip和vtkdata-X.X.X.zip文件解压到你的硬盘中，在VTK源代码文件夹下，可以找到“Examples”文件夹，文件夹“Examples”里又有类似“GUI”，“MangleMesa”以及“Parallel”的文件夹，每个文件夹的子目录里都会有一个名为“Tcl”的文件夹，里面包含了大量的Tcl例子。除了“Examples”文件夹，还有大量的库目录，像Graphics, Imaging和Filtering。每个目录下都会包含一个Testing/Tcl子目录，里面有对VTK逆行测试的例子。双击这些Tcl文件即可运行，在你还没有安装Tcl/Tk时，初次双击后缀为.tcl的文件时，会弹出对话框，询问你用什么程序打开文件，这意味着你要把Tcl文件与vtk可执行文件做一下关联，也就是在“打开为…”对话框中选择“浏览”，定位到VTK安装目录下，通常是“C:\Program Files\VTK 5.6”或“C:\Program Files\Documents and Settings\<username>\My Documents\VTK 5.6”，然后选择bin目录，选择可执行文件vtk.exe，可以把“打开为…”对话框中的“使用此程序打开此种类型的文件”复选框选上，再点击“确定”按钮，以后双击.tcl后缀的文件就可以用VTK自动运行Tcl脚本文件了。



图2-1 Windows下VTK安装程序界面

另外，如果你已经安装了Tcl/Tk二进制文件的话，\*.tcl的文件会自动与Tcl/Tk程序进行关联，这时你再双击\*.tcl文件时要出现如下的错误提示：can’t find package vtk while executing “package require vtk”。可以用以上介绍的方法，把\*.tcl文件与VTK可执行程序进行关联。

以上就是在Windows系统下安装VTK二进制文件的全部过程，在第三章里我们会详细介绍如何书写自己的C++类，以及Tcl, Java和Python应用程序。

**源代码安装**

如果你打算用VTK开发C++应用程序以及扩展VTK的话，就有必要采用源码安装方式。这种安装方式可能是一件比较有挑战的事情，它与你计算机的配置有很大关系，可能需要花费你不少时间。首先要确保你的机器适合编译发布的VTK源码。同时你的机器也应该安装了Windows XP, Vista或以上版本的操作系统，以及必须安装了C++编译器，这个指南是以Microsoft Visual Studio 2005及更高版本的编译器做测试的，VTK可以很好的用这些版本的编译器进行编译。同样也支持Borland C++编译器、Cygwin下的gcc编译器或MinGW，NMake，Microsoft Visual C++免费版本以及Microsoft Visual C++ 2005等。如果你机器上一个编译器也没有安装的话，先安装以上所说的编译器之一(注：推荐安装Microsoft Visual Studio 2008)。

接下来你就要考虑还需要使用其他什么工具。如果你准备用Java开发程序的话，必须下载并安装Java JDK(可在Sun Microsystems: <http://www.java.sun.com>上下载到)。如果你打算用Tcl/Tk开发，必须下载并安装Tcl/Tk(源码可从<http://www.tcl.tk>下载，或者从<http://www.activestate.com/Products/ActiveTcl>下载Tcl/Tk的二进制安装文件)。注意，Tcl/Tk 8.4版本可以与VTK 5.4.0版本协同工作。

**安装CMake**。要编译VTK，首先你必须先安装CMake。CMake的安装程序可以从<http://www.cmake.org>上下载到。

**运行CMake**。安装完C++编译器、CMake以及你需要用到的其他工具后，如Tcl, Java, Python，就可以运行CMake程序。打开CMake程序，界面如图2-2所示。CMake会根据不同的编译器以及操作系统生成VTK项目文件。在”Where is the source code”一栏中指定VTK源码所在的目录(也就是压缩包vtk-5.6.1.zip解压后所在的目录)，在”Where to build the binaries”一栏指定VTK编译过程中生成的文件的目录(最好建立一个空的文件夹，这样不会对VTK源码造成污染)，这两个路径的设置可以通过右边的”Browse Source…”和”Browse Build…”按钮进行选择，不用手动输入。设置完这两个路径以后，就可以按”configure”按钮。初次configure时，会弹出对话框让你选择编译器，根据你机器的安装情况选择合适的编译器。接着CMake会自动填充变量列表的值。第一次运行configure以后，所有的变量条目会以红色显示，红色标示着缓存条目被自动生成或者在上一次configure时值被作了修改。

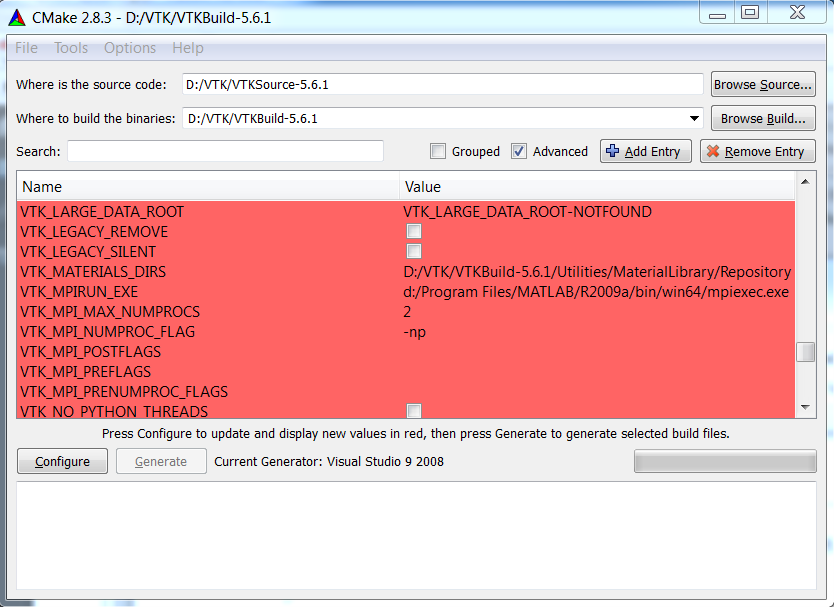


图2-2 CMake会根据不同编译器、操作系统生成工程文件、Makefiles等，CMake是跨平台的，图中所示为CMake图形用户界面。

Configure完了以后你就可以定制VTK编译的设置选项。比如，如果你想激活VTK的Tcl封装特性，你就必须把VTK\_WRAP\_TCL这一项勾选上(设置其值为ON)。设置完这些选项的值以后，再次按Configure按钮，直到所有的选项都变成了灰色。如果你安装了Tcl/Tk的二进制文件以后，关于Tcl/Tk的一些选项还是显示NOTFOUND的话，你就必须手动设置它们的值。可点击每个选项值右边的”…”，在弹出的对话框中选择合适的文件或目录。

下面列出VTK中一些重要的变量值：

* BUILD\_SHARED\_LIBS – 如果设置成yes(即勾上了这个选项)，VTK就会编译出一些DLL或者说共享库文件，如果设置成no，则编译成静态库文件。缺省值是no，即编译出来的链接库是静态的。某种程度上，静态库可能比较好使用，因为在可执行文件运行时，不要求这些库文件在当前目录下或系统Path环境变量下，可执行文件会自已包含相关的链接库。这对于发布基于VTK的程序也是比较有利的。
* VTK\_WRAP\_TCL – 这个选项决定是否编译Tcl的封装特性。
* VTK\_WRAP\_PYTHON – 决定是否编译Python的封装特性。
* VTK\_WRAP\_JAVA – 决定是否编译Java的封装特性。

设置完相关选项的值以后，再次按Configure按钮，直到所有的选项都变成了灰色。然后按”Generate”(有些CMake版本可能是”OK”按钮)，CMake就会根据选择的编译器类型自动生成一些工程文件。如果是Microsoft Visual Studio，就会在”Where to build the binary”指定的目录下生成一些工程文件(Visual Studio 6.0是vtk.dsw文件，VS2005或更高版本的是vtk.sln文件)。在Visual Studio环境中加载工程文件后，选择编译版本(Debug, Release, MinSizeRel, RelWithDebInfo等)，然后选择ALL\_BUILD工程，编译工程。如果是使用Borland编译器，则生成的是Makefiles文件，你必须以命令行的形式给编译器指定合适的参数。Makefiles文件也是位于”Where to build the binary”指定的目录下。

当所有的VTK工程都编译完成以后，所有的库文件以及可执行文件就会出现在CMake里指定的编译目录下的子目录bin下(除非你更改了CMake里的选项EXECUTABLE\_OUTPUT\_PATH或LIBRARY\_OUTPUT\_PATH选项的值)。

(注意：不要使用MSVC++中的“Rebuild All”菜单来编译源代码，这个过程将删除所有CMakesList.txt文件，然后在编译过程中自动重新生成。MSVC将试图重新加载他们，这样就会导致错误发生。要重新生成每样内容的话，清空VTK编译目录，返回到CMake，按以上的步骤重新生成工程文件，然后再编译。)

如果选项BUILD\_SHARED\_LIBS的值设为ON的话，VTK程序运行时要加载这些DLL文件。这有不同的方法可以让VTK程序运行时加载DLL库文件，每种方法都有利有弊。最简单的方法就是保证你的程序以及VTK的DLL库文件在同一个目录里，你可以复制所有的VTK DLL库文件到你的应用程序所在的目录去，或者把你的程序的编译目录跟EXECUTABLE\_OUTPUT\_PATH指定的路径是一致的。一旦所有的VTK DLL文件以及你要执行的应用程序都位于同一个目录，一切就会正常了。这种做法的好处就是它简单，不好的地方就是一旦你更新了VTK DLL文件以后，你就不能更新之前复制过去的VTK DLL文件，除非你把重新编译的VTK DLL文件再复制一遍；而且一旦你的应用程序的编译目录与VTK的DLL文件混在一起的时候，可能会导致应用程序运行加载文件混淆，不知道应该加载哪个。

另一种方法就是改变PATH环境变量(也就是把包含编译好的VTK DLL文件所在的路径放在系统环境变量PATH里，右击“我的电脑”->“属性”->“高级”->“环境变量”进行设置)，这样尽管你的应用程序与VTK DLL库文件不在同一个目录下也能正确找到要加载的动态链接库。

KWWidgets提供了一个使用批处理脚本文件设置PATH变量(不用去修改系统环境变量PATH)的很好的例子。可参考KWWidgets目录下的KWWidgetsSetupPaths.bat脚本文件，KWWidgets源代码文件可从<http://www.kwwidgets.org>下载。

如果想了解更多关于Windows下DLL文件加载方面的内容，可以参考Windows SDK文档里的LoadLibrary和LoadLibraryEx函数。

至此，就成功地在你机器上安装了VTK，因为VTK的复杂性以及含有大量的源码文件，可能使得这个过程具有一定的挑战性。如果在这个过程中你遇到了什么问题，可以在vtkusers邮件列表上提出(参考第一章“其他资源”一节)以获取帮助。

## 2.4 Unix平台下安装VTK

Unix系统存在大量的版本，因此你不得不通过编译VTK源代码来建立二进制和可执行程序文件。

**源代码安装**

这部分内容将引导你过一遍在Unix系统下编译VTK所需要的步骤。和Windows系统不同，预编译的二进制和可执行程序在多数Unix系统上是不可用的，因此你必须自己编译VTK。（注意：查找一下vtkusers邮件列表及第一章“其他资源”一节介绍的资源，你会发现有些用户已经在Web上提供了编译过的二进制文件。）一般来说，这个过程还是相当简单的，根据你机器的配置情况可能会花上一到四个小时。（高端、大内存、多处理器的机器使用并行编译可以在十分钟内编译完C++和Tcl的二进制和可执行程序！）绝大部分的时间都花在等待计算机编译源代码上。只需要你10到30分钟的时间，第一步是确认是否有编译VTK所必须的资源。为保险起见，你最好空出300M的磁盘空间。某些系统，如SGI，你得留出更多的空间，特别是如果你要编译VTK的调试版本。由于VTK是使用C++编写的，你还需要C++编译器。一般的C++编译器为CC、g++或者ACC。

如果想在Tcl/Tk、Python或者Java中使用VTK，那么你需要先下载和安装这些程序包。Java JDK可以从Sun Microsystems <http://ww.java.sun.com>获得；如果你打算使用Tcl/Tk，你需要下载并安装Tcl/Tk，可以从<http://www.tcl.tk>获得；Python 可以从<http://www.python.org>下载。下面介绍如何编译他们。

**CMake**

与Windows环境类似，在Unix系统下编译VTK也需要借助CMake(见前面关于CMake介绍部分)。Unix系统下，有多个版本的CMake预编译二进制文件可以使用，如果这些版本的CMake都不可用的话，你还必须先编译CMake。(可从http://www.cmake.org下载到CMake的二进制安装文件)。

**安装CMake**。如果二进制安装文件不可用的话，下载CMake的源代码tar文件，解压到目标目录下(一般是/usr/local/)。要确保cmake以及和它关联的可执行文件都在当前路径下，或者给出完整路径运行cmake。

**编译CMake**。如果CMake的二进制安装文件不可用的话，你就必须自己编译安装CMake了。要编译安装CMake只要把CMake的源码包(可从<http://www.cmake.org>下载到)解压到一个目录中，然后在那个路径下运行下面的命令即可：

./configure

make

make install

如果你没有root权限，就得省略上面的第三个步骤(也就是make install)。CMake可执行程序位于CMake/bin目录下。CMake提供了两种不同的可执行程序文件来配置VTK：ccmake提供了一个基于界面的终端，非常类似Windows系统下的cmake-gui；以及cmake向导，需要你一步一步地回答一系列问题以完成配置过程。

你还得告诉CMake采用何种C++/C编译器。在多数系统中，你可以通过下面的方式设置相关的信息：

setenv CXX /your/c++/compiler

setenv CC /your/c/compiler

或者

export CXX=/your/c++/compiler

export CC=/your/c/compiler

否则CMake会自动检测你编译器，如果你机器上安装了多个编译器，CMake检测到编译器未必就是你想使用的。完成了这些后，在VTK源码的同级目录中创建一个空的目录(比如叫VTK-bin)，然后在这个目录下运行CMake，把你刚才创建的目录以及VTK的源码目录的路径作为参数传递给CMake。

cd VTK-bin

ccmake ../VTK

或者

cd VTK-bin

cmake –i ../VTK

(接下来两小节的内容会介绍ccmake和cmake –i的区别)。对配置脚本比较熟悉的Unix开发人员会注意到CMake和configure在功能上是类似的。但是configure是需要命令行参数，而CMake是可以通过用户交互设置编译选项以生成makefiles。

**用户交互终端定制编译选项(ccmake)**。ccmake有一个基于用户交互的简单的终端，允许你根据特定的机器以及定制选项设置VTK的编译环境。当你用ccmake运行CMake时，会有一系列选项供你选择、修改，以便你根据特定环境设置VTK的编译环境。CMake缺省情况下会设置绝大部分选项的合理值。可以通过键盘上的箭头按键更改每个选项的值，当理想的值高亮显示时按回车键即为选中。也可以编辑选项的值(除了该选项是布尔类型的，如果是布尔类型的选项，按回车键可以更改它的取值)，完成编辑的选项值以后再回车以确定该选项的值。当所有的选项值都设置完成以后，按”c”键即可完成整个设置过程。接着CMake就会进入处理配置文件的过程，必要时，CMake还会在顶端部分显示新的选项(比如，如果你打开了选项VTK\_WRAP\_TCL，就必须再设置Tcl/Tk的库文件以及包含的路径)。如果有新的选项出现，应该重新设置它们的值(如果这些新的选项有合理的缺省值时，也可以采用这些缺省值)。然后按”c”键重新配置，直到没有新的选项出现。这个过程完成以后，就会有新的可用的命令出现：”Generate”和”Exit”。你可以按”g”键让CMake生成makefile文件和退出。以后如果你还想重新改变某些选项的值的话，只要根据以上步骤再次运行ccmake即可。

**向导模式定制编译选项(cmake –i)**。某些平台下，ccmake提供的用户交互终端可能不好使，这时可以尝试键入带-i选项的cmake命令(激活向导)。向导会询问你是否使用”高级选项”，大部分用户没必要使用高级选项。接着CMake会一步一步地询问你每个选项的取值，每个选项都会列出它们的详细描述信息以及缺省值。大多数情况下，CMake会自动生成可以接受的缺省值。当然，某些情况下，比如OpenGL库文件没有位于预期的位置时，你就必须告诉CMake确切的位置。另外，缺省情况下不会创建对Tcl, Python和Java的绑定。如果你想支持这其中的一种语言的话，应该打开选项VTK\_WRAP\_XXX，必要时还要告诉CMake相应库文件和头文件的位置。回答完CMake的所有问题以后，CMake就会自动生成makefile文件，VTK就可以开始进行编译了。以后如果你想更改某些选项的值的话，可以根据以上步骤再运行CMake向导。

**编译源代码**

运行完CMake后就会生成makefile文件，可以键入make命令，开始编译VTK。一些make工具，如GNU make(gmake)支持并行编译(如, gmake –j)。尽可能使用并行编译，即使在单处理器系统，因为通常这个过程是IO瓶颈，处理器可以处理多个编译。如果你在这个过程遇到什么问题，可以到vtkusers邮件列表(参考第一章“其他资源”一节)寻求帮助，也可以从Kitware获得商业支持。

**多个平台下编译VTK**

如果你打算在多个平台下编译VTK，可以根据上面所讲的为每个平台复制一份完整的VTK源文件目录，或者你也可以复制一份VTK源文件目录，然后在另外一个目录下为每个平台生成目标代码、库文件以及可执行文件。这种方法需要为每个平台创建一个空的编译目录，比如vtk-solaris(确保你的硬盘有足够的空间)。假定VTK的源文件目录跟你所创建的空的编译目录位于同一层目录里，改变当前目录(cd)到这个新建目录中，按照下面类似的方式运行CMake：

cd /yourdisk

ls （输出是VTK vtk-solaris vtk-sgi）

cd vtk-solaris

cmake – i ../vtk

或者

ccmake ../VTK

就会在目录vtk-solaris中生成makefile文件。这样你就可以根据上面提到的步骤编译VTK。

**安装VTK**

现在VTK已经编译好了，可执行文件和库文件都会在编译目录的子目录bin/中。如果你打算在UNIX系统中和一个以上的开发者共享这些库文件，而你又有root权限的话，可以运行make install命令，这将会把VTK安装到/usr/local中（除非你改变配置选项中的CMAKE\_INSTALL\_PREFIX，将其设置为另外的位置）。运行make install命令将会复制你编译和运行VTK应用程序所需要的所有文件到一个其他用户都能使用的目录下。

这就是在UNIX下编译和安装VTK的完整步骤，如果你想了解更多关于CMake的内容，可以登录http://www.cmake.org或者购买Kitware公司出版的图书《Mastering CMake》。第3章将会详细讲解怎么运行VTK例子以及创建自己的应用程序。

翻译者：罗火灵

QQ: 348774226

Email: shuiling119@hotmail.com