# 目录

# 1. 介绍

正如世界上每种事物都有其发展趋势,今天的计算机看起来正越来越倾向于对免费软件的使用,甚至连商业用途的软件也不例外.最有名的免费软件当属 Linux.目前人们普遍认同,Linux(和其他一些项目,例如 Apache 网络服务器,Perl 语言与 GNU 工具箱)证实了免费软件也可以象商业软件一样保证很高的质量.但是除开质量问题不谈,终端用户仍然必须忍受任何 unix 系统的苛刻的命令之苦.Linux 系统要想更加繁荣,就需要不断应用,不管是免费软件还是商业软件;以及简单易用性.

KDE 工程尽力通过提供对桌面的方便使用和自带的函数库来增加可用的基于 GUI 的软件资源的种类,来填补这一空白.但是尤其因为免费软件通常都是作者利用空闲时间做成的,对许多程序员来说,问题的关键就在于他们对现有的代码开发环境的喜爱程度.Kdevelop 想要采取另一个重要步骤:让程序设计者的生活更加简单而高效:用 Kdevelop 开发出来的产品可在同样长的开发周期内在可靠性和功能上达到一个更高的水平。

为了实现这一目标,Kdevelop 集成开发环境提供了很多程序开发者需要的特征,同时它也集成了第三方项目的函数库,例如 make 和 GNU C++ Compilers 编译器,并把它们做成开发过程中一个可视化的集成部件. KDevelop 管理:

- 所有开发 C++程序所需要的开发工具, 例如 Compiler 编译器, Linker 联接器, automake 和 autoconf ,
- KAppWizard 应用向导, 可生成完全的, 可执行的应用实例,
- 类生成器, 可产生新的 classes 类并把他们集成到当前项目中,
- 文件管理器, 管理源文件, 头文件, 文件等使之被包含在项目中,
- 用 SGML 编写的用户使用手册的创建和由 KDE 自动生成的 HTML- 输出 output,
- 为你的工程的各个 classes 类自动生成的基于 HTML 的 API-documentation 文件, 在以前的库中可查,
- 为您的运用提供全球化支持, 允许翻译者将其目标语言轻松地加入一个项目,
- 所见既所得, ---由内置的对话框编辑器 dialog editor 完成的用户界面的创建,
- 利用 CVS, 通过为最急需的函数提供一个易于使用的前端来管理你的项目,

- 通过集成 KDbg 调试你的应用程序,
- 利用 KIconEdit 编辑项目细节 pixmaps，
- 根据你自己的个人需要,你可以通过"Tools"菜单把其他任何你所需要的开发程序包括进来。

KDevlop 使得在一个地方同所有的程序工作成为一种乐趣,它通过自动的标准开发程序,和提供你对所有所需信息的直接路径来节省你的时间.集成浏览器的设计支持开发者在连接中对其项目提出文档 documentation 请求。

classs viewer 类浏览器和错误寻找器通过鼠标点击,无需寻找文件就可以把你带到项目代码的任何地方;文件树给你到达 project files 项目文件的直接路径,集成的帮助系统也通过 IDE 提供从任何地方到达在线文档的超级路径。

## 1.1 改变

KDevelop1.0 版本拥有开发 C/C++最需要的设施。在 KDvelop 开发阶段,项目从 0.1 以及随后发布的 0.2,0.3 和 0.4 版起,已经有了很大的提高,他们中的每一个都经过了检验和固定调试. 我们相信,目前的版本将成为 KDE 1.x 系列的最终发布版本,会给开发者们一个他们希望使用的开发环境.

改进主要在以下一些部分:

- 快速的类语法分析器和显示图表、以树的方式浏览类的浏览器和基于对话的成员及类的加入,
- 对最常用的的功能--例如增加，去除，升级，及提交文件及目录--的 CVS 支持,
- 更新及扩展过的文档， 现在随 KDevelop 一起发布的共有 5 本手册,
- 错误定位类，语法分析器， 对话框编辑器和错误寻找器。

任何有关 IDE,其文档以及特色要求的评论都会受到欢迎.

## 1.2 关于这本手册

这本用户手册提供给用户一个 KDevelop IDE 的完整的总体概况,并简要描述了其基本开发过程.要想得到关于它的更多信息,我们建议您阅读 KDevelop 自带的 KDevelop 编程用户手册 KDevelop 编程手册,它包含了例如如何理解生成的应用框架 application frameworks 及如何使用项目实例生成功能完全的 KDE 应用的

话题。

因此这本手册的设计被分为以下几部分:

- 安装篇, 包括系统要求,安装和 KDevelop IDE 的安装。
- 编程篇, 告诉你程序是如何由标准的 GNU 开发工具生成和组建的。
- 利用 KDevelop 开发篇 , 简要说明开发环境的主要功能。
- 概要篇, 解释菜单、对话框和键盘热键 shortcuts。
- 帮助系统篇, 告诉你如何使用内置的 documentation browser 文档浏览器和另外的帮助功能。
- 编辑篇, 说明文件管理和编辑的特点。
- 项目篇, 描述软件项目的产生和支持。
- 编译篇, 包括关于如何设置 Compiler 编译器、连接器标签和项目选项的说明。
- 类浏览器篇, 告诉你如何使用 KDevelop 中功能强大的类浏览器。
- 对话框编辑器篇, 描述集成的可视化 GUIconstructor/以及它如何生成 C++output 输出的。
- 一般设置篇 说明如何为使用 KDevelop 进行全局设置。
- 问与答, 包括了在不同风格的 Unix 系统中使用 KDevelop 中会遇到的问题和问题的一般解决方法。

对于程序设计新手和才开始使用本产品的用户,我们建议您在正式开始使用 IDE 之前首先仔细阅读这本手册,它介绍了详细的使用方法.对事物运作情况的深刻理解是节省你寻找各个功能和特色的时间的最快方法,因为它让你能够使用虽然更简单却是第一流的程序开发工具。

## 1.3 关于 KDevelop 项目

KDevelop 项目开始于 1998 年的夏天,也就是 KDE 1.0 发布以后. 我们希望为程序员，特别是为 KDE 桌面开发应用的程序员提供一个简单易用的 C/C++集成开发环境.之后,该项目得到了许多人的帮助和支持,他们和项目的发起者一样希望帮助开发组继续这个被广泛认可的编程环境.在一年半内,该 IDE 发展成为一个功能齐全而稳定的开发环境,以及很好的可用性,因为开发者们使用这个 IDE 来开发自身.因此程序的错误以及编写 KDE 应用程序必须的扩展往往被作者自己发现,而错误则并很快得以更正.

第一个于 1999 年 9 月 6 日正式发布的稳定版本是 1.0 版.从那以后,KDevelop 小组通过添加新代码和修改以前不稳定的部分以不断的向 IDE 中增加新的特征和加强稳定性,从而导致在 2000 年 2 月 28 日发布了第二个版本,即 1.1 版

从此,该项目不断的集成更多的功能并由许多翻译者将 KDevelop 译成尽量多的语言给国际开发者提供他们本国语言的集成开发环境,包括附带的手册.许多使用者把他们的经验和知识贡献给这个 Linux/Unix 下极有希望的项目产品,以使其具有更好的稳定性和对其它 Unix 平台的良好移置性.

## 1.4 可打印的版本

正如标题所示,KDevelop 完整的文档已经有了可打印的版本"**为 Linux 开发应用程序:KDE 版本**",你可以从 KDevelop 主页 http://www.kdevelop.org 或者 http://www.opendocs.org 订购. 销售该书的毛利将被返回给 KDevelop 项目一支持其后续版本的开发

KDevelop 文档的可打印版本也可以从 KDevelop 主页 http://www.kdevelop.org 获得,以 DVI,PostScript 和 PDF 格式,A4 和信纸大小提供

# 2. 安装篇

## 2.1 如何获得 KDevelop

KDevelop 可以在地址为 http://www.kde.org/current.html 的 KDE 应用页面上或者地址为 http://www.kdevelop.org 的 KDevelop 主页上找到. 在 Linux 的发布中也可以获得, 例如 SUSE 6.1 版.

我们还在我们的主页上为那些希望获得最新版 KDevelop 的用户提供 KDevelop CVS 库的快照.一般来说,这些快照并非着意于被用在产品开发上,而是作为对新特色的测试并使大家了解 KDevelop 小组的开发进度.我们还直接提供 KDevelop 所需要的各种第三方软件,例如 KDoc 和 KDbg。如果您现在遇到了编译或使用 Kdevelop 的问题,请阅读这本手册的 问与答 部分或 KDevelop 包里的 FAQ 文件.如果您遇到的问题没有叙述到,请将您的问题通过一封头部为空,以 "subscribe" 为内容的信件提交至地址为 kdevelop@fara3.cs.uni-posdam.de 的 KDevelop 邮件表单.要求和问题报告应仅以 KDevelop IDE 的使用为目标,而不是关于您在编写自己的应用程序中所遇到的任何实现 implementation 问题.不管怎样,所有送至邮件表单的信件都应该用英文书写, 这样所有的参与者都可以加入讨论,这样也能为您提供更好的帮助.邮件表单也着意于那些愿意为此作出贡献或已经找到自己所遇到问题的解答方法的用户,这样我们也能确定 errors 错误并将这些知识给予那些初学者一个更专业的第一手帮助.报告问题的一个好方法是通过从控制台启动 KDevelop 寄出你得到的 output 输出,或粘贴 KDevelop 的内部 Messages-window 信息窗的内容。

## 2.2 要求 Requirements

为了成功编译和使用 KDevelop,你需要以下程序和库,它们在大多数平台上作为发布的软件包都易于得到,所以安装起来也很容易。

**必要项:**

- g++ 2.7.2/g++ 2.8.1/egcs 1.1 (或可兼容), 在地址 http://www.gnu.org 可以获得。
- GNU make (或可兼容), 在地址 http://www.gnu.org 处可以获得。

- perl 5.004, 在地址 http://www.perl.com 处可以获得。
- autoconf 2.12, 在地址 http://www.gnu.org 处可以获得。
- automake 1.2, 在地址 http://www.gnu.org 处可以获得。
- flex 2.5.4,
- GNU gettext, 在地址 http://www.gnu.org 处可以获得。
- Qt 1.42, 在地址 http://www.troll.no 处可以获得。
- KDE 1.1.x, 在地址 http://www.kde.org 处可以获得。

**可选项:**

- a2ps 或 enscript 以提供打印 printing 支持。
- ghostview 或 kghostview 以提供打印 printing 预览。
- glimpse 4.0 以提供搜寻索引 search index, 在地址 http://glimpse.cs.arizona.edu 处可以获得。
- sgmltools 1.0, 在地址 http://www.sgmltools.org 处可以获得。
- KDE-SDK (KDE Software Development Kit), 它包括了 KDoc, KSgml2Html, KTranslator(在地址 http://developer.kde.org 处可以获得。)
- KDbg, 在地址 http://members.telecom.at/~johsixt/kdbg.html 处可以获得。
- KIconEdit (在地址 http://www.kde.org 处可以获得。)

KDevelop 已在装有 SuSE Linux 5.2 系统、AMD k4200、64MB RAM 的机器和装有 SuSE Linux 6.0、Intel 200 MMX、128MB RAM 的机器上经过测试。

就作者所知，SuSE Linux 和 FreeBSD 包含了所以必需的软件包，其中有 a2ps 和 enscript 软件包，这样您就不会遇到安装第三方软件的问题了。

**文档:**

为生成 KDE 库文件,您需要 kde 库软件包,这在你系统中作为 KDE 项目提供的资源里或在你软件和 **Kdoc** 的资源软件包里可以获得。(包含在 KDE-SDK 中).

我们还在我们的主页上提供一个安装后集成在 documentation browser 文档浏览器中的 C/C++Reference 参考,主页地址为 http://www.kdevelop.org.下载软件包后将 root 源文件拷贝至您的 KDE 目录中，并用 tar zxvf c_c++_reference.tar.gz--------- 它.这样这一参考就在文件树中可见了;否则在浏览

器中选择这一参考书会出现一个错误的页面,这是 Kdevelop 的主页,它提供下载并叙述了安装过程.

## 2.3 编译和安装

为了在您的系统上编译和安装 Kdevelop,请打出 Kdevelop 发布软件的基本目录中的以下内容:

```
% ./configure
% make

(作为超级用户)

% make install
```

Kdevelop 使用了 autoconf,因此您编译起来就不费力气了。

为了编译 Kdevelop CVS 快照，请打出以下内容 ：

```
% make -f Makefile.cvs
% ./configure
% make

以超级用户身份：

% make install
```

如果您的系统的 make-command 是 gmake,打出 gmake 而不是 make。

## 2.4 启动 KDevelop

如果您使用 KDE 管理您的窗口， Kdevelop 可以通过选择"K"->"Applications"->"Kdevelop 1.x"来启动。Kdevelop 支持 KDE-Mime-types, 因此您还可以通过选择一个 Kdevelop 项目文件(".kdevprj",由 KDEvelop 项目图标显示。如果使用的是其他窗口管理程序，请打开一个控制台并打出：

```
% kdevelop
```

为使 kdevelop 启动时打开一个以存在的工程，请将目录变为项目目录并打

出：

```
% kdevelop  <yourProject>.kdevprj
```

在每一个用户帐号下 Kdevelop 都会在第一次使用时请求 安装程序 自动安装程序，允许对最需要的选项进行快速设置。如果您的安装出现混乱，您可以在任何时间进入时重新设置。

```
% kdevelop  --setup
```

或者，当您使用 KDE 时,通过选择"K"->"Develop"->"KDevelop Setup"。

## 2.5 安装程序

Kdevelop 有一个自动安装程序模块，无论何时当 Kdevelop 被启动时就可以把它激活，而 kdevelop 设置文件并不存在。我们建议您在安装 Kdevelop 环境时遵循安装步骤--选择" 进行"选项以自动检查您的系统及建立你的 Kdevelop 环境。

安装对话的按钮需要执行以下步骤：

**帮助**：这会启动 KDE 帮助程序。

**进程**：这将启动安装程序并执行以下步骤：

1. 检查 make/gmake, autoconf, autoheader, automake 和 perl 以生成和编译 Kdevelop 产生的新的应用程序。如果安装了 gmake,make-command 会在使用 gmake 时被自动安装。其他编程时用到的命令行选项不久后可在 setupdialog 中由选项菜单设置，在 Unix 环境下的开发说明会在 编程篇 中介绍。
2. 检查 KDoc 和 glimpse. 如果找到，会允许在下一步生成一个新的 KDE-Librarydocumentation 文档和 search index 搜寻索引。
3. 检查 a2ps and enscript 以保证 printing 打印可用。这些程序的任何一个都必须安装以保证正常 printing 打印。如果任意一个未安装，你可以在以后的任一时间在你的选项中完成它而无需再次运行 setup 安装程序。
4. 检查 KDbg, KIconEdit and KTranslator. 我们鼓励您安装这些程序，他们是生成一个完整的 KDE-applications 应用程序的好帮手。注意

到 KDbg 在 Kdevelop 中被直接用于 debugging 调试你的当前项目；KiconEdit/被用于表示和编辑文件浏览树中选中的 pixmaps。如果被发现，KDbg 程序， KiconEdit 程序和 Ktranslator 程序会在 Kdevelop 菜单条的 Tools 工具菜单中被设置为可用。其他工具可以在其后通过选择选项菜单中的 Tools...被加到 Tools-menu 中。

5. 检查程序的总结：安装程序列出它所发现的和未知的程序。如果要推荐一个需要的程序，我们会给出特别的暗示。

6. 检查 Qt-Documentation path 文件路径：它将检查你的系统中用于文档的几个标准的路径，并自动设置路径。如果你的 Qt-Documentation 文档由于未安装，或者你的系统把它放在另一个位置而没有被检测到，信息框会被弹出以询问你是手动设置正确路径还是继续。选择手动设置按钮会返回主安装窗口，并显示一个编辑区域，右置按钮以选择路径。特别的是，这是在 qt/html 目录中的。然后，再次选择"Proceed"选项，安装过程可以继续下去。

如果 KDoc 被检测到，系统会询问你是产生还是更新你的 KDE-library documentation 库文档。为此你需要将 kde-libraries 作为资源。对已从发布的软件安装了 KDE 的 Linux 的使用者，我们建议你将你系统中的 kdelibs 软件包资源拷贝并解压；而安装了 kdelibs 软件包的免费 BSD 的用户则应该查看他们的相关文件包。如果你并非以上两种情况，你应该从地址 http://www.kde.org 下载这些资源并将其安装在你的系统中。如果你想使用 http://developer.kde.org 提供的文档包，请取消生成此文档，继续产生 search index 搜寻索引。安装程序完成，而且 Kdevelop 已经启动以后，在 Kdevelop 安装程序对话框中给解压文档包设置路径，并再次运行 search index 搜寻路径及 setup，这可在对话的同一 setuppage 页面中找到。 "更新 KDE-Library documentation/库文件"对话框为使用缺省路径定位 $(HOME)/.kde/share/apps/kdevelop/KDE-Documentation 中的文档而设置。你唯一需要做的一件事就是在对话框中选择顶部的选择按钮，选择到达你在系统上展开的资源库的路径并按下 OK。例如，如果你是从 http://www.kde.org 处得到的 kdelibs.tar.gz 软件包并把它下载到了本地目录，你需要启动一控制台或计算机终端并进入"tar zxvf kdelibs.tar.gz"。这会将资源放入目录$HOME/kdelibs，那时它的每一子目录的库中已包含了资源，例如 kdecode，它会在 / home/rnolden/kdelibs/kdecore。现在，你必须输入 KDE 库文件对话框的路径会成为所有库的路径，在例子 / home/rnolden/kdelibs 中。在按下 Ok 按钮以后，安装窗口的信息框会表示文档正逐步生成，你需要等待下一信息。

**注意**：在多用户系统 multi-user system 或为用户帐号有 disk-quotas 的系统中，每一个用户 HTMLdocumentation 文档的完全安装是对磁盘空间的极大浪费。在此情况下，应该请你的系统管理员在根部帐号下 under the root account 运行 Kdevelop，允许在系统根目录下进行写操作。然后在 KDE-directory 目录里安装文档 $KDEDIR/share/kdevelop/KDE-Documentation。然后正确的路径可以随后在 Kdevelop 的 setupdialog 中建立，由"Option"-menu 进入。

7. 如果系统检测发现你的系统上有 glimpse 程序，你可以建立一个查找数据库。查找数据库可用选项建立，包括 KDE-Documentation 和 Qt-Documentation(缺省)。此外，KDEvelop 文档会被包括入内并编入索引。如果你还有其他你希望索引的文档，你可以选好目录并将它们也加入索引程序。

8. 在建立 search index 搜寻索引时，安装窗口会弹出一个信息框，说明安装正在进行。

9. 如果所有安装步骤都已正确执行，最后一个信息框会告诉你 Kdevelop 会在你按下 OK 后启动。注意：你可以选择选项菜单进入 Kdevelop Setup 安装对话框，设置另外的选项，例如自动存盘。

**取消**：会弹出警告框，说明安装过程将被取消。这一警告框允许你选择返回安装（"返回"）或者在缺省参数下启动 Kdevelop（"继续"）。注意：这样你就必须利用选项菜单提供的 configuration dialogs 配置对话框自己设置所有的选项。

# 3. 程序篇

现在 Kdevelop 已经正确安装完成，最常用的选项也已设置完毕，你可能很想知道它是否如它所承诺的一样好用。本章将就程序是如何由 GNU 工具创建的作出总体指导，尤其要说明 Kdevelop 在此过程在中所起的作用。

## 3.1 编译器

Compiler 实际上是你系统上的一个程序，它必须最小安装以创建运行的程序；它是将源代码编译为目标文件，并生成程序的部件。一般来说，你可以这样开始：打开一个你喜欢的编辑器——注意不要使用文字处理器。打出例如以下字段以创建你的第一个程序的源文件：

```
#include <iostream.h>

int main(){

cout << "Hello World" << endl;

}
```

好，实际上程序所有要做的事就是以标准 output 输出打印出字符串"Hello World".但这仅仅是建立这个程序的源代码，而并非程序本身。因此，我们需要一个 Compiler 编译器，在此例中是一个 C++-Compiler 编译器，例如 g++。这样我们就可以将源代码保存为文件例如 myprogram.cpp，并可由文件名(在控制台上)激活 Compiler 编译器:

```
g++ -o myprogram myprogram.cpp
```

然后我们就可以启动我们的程序了——只需在控制台上打出 myprogram，程序就会打印出字符串；然后退出。

## 3.2 Make 和 Makefiles

我拥有我所需要的一切东西：一个编辑器，一个 Compiler 编译器，然后我就可以执行我自己的 C++程序了。但事情并非总是那么容易。如果你有一个以上的源文件怎么办呢？当你仅改动了一个程序时，你需要一次又一次的重新编译所有的源文件吗？由于你必须打出所有的命令和选项，编译会变得越来越复杂和费

时。为此，你可以先编写好一个所谓的" Makefile"。你可以给它取任何名字，但不要和你要建立的程序名字一样。然后，你应该安装好 make 或者 gmake 工具，或者可以保存某个项目的编译结果的任何其他工具。把你的所有 Compiler 编译命令以一定句法写入哪个 Makefile 并保存；然后你只需于控制台上打出 make 或者 gmake 到你的 Makefile 文件所在的目录，再确定任务，指挥 Compiler 来创建你的应用程序。"创建"工具还有很多其他优点，有很多功用。如果想了解完整概况，请打开一个控制台并打出：

man make

或在 KDEHelp 中搜寻"GNU Make"，"System GNU Info contents"。至少，你已经认识到，为什么一个开发者需要 make 工具来让使他的应用程序的编译更容易。现在，编写 Makefiles 不仅仅是一项手动工作，你还必须钻研所有整个句法和所有选项。但这里有一个关于 Kdevelop 和任何 Make-utility 的好消息：你只需在 Kdevelop Setup KDE 一般设置对话中设置好 Make-命令就可以了。所有由 Kdevelop 创建的项目都会使用那个 Make 命令来建立目标应用程序，而无须打任何命令。只要在 KDE 的工具条上点击按钮，从第三条分隔线开始，或者在"Build"菜单中为 Make 选择想要的工具。然后工具条和 build 菜单会提供你需要的最常用的功能来由 make 完成这个累活。

- Compile File 编译文件：只有当你正着手于一项源文件时才是激活的。它用正确的命令激活 make 来编译当前源文件。
- Make 建造：访问 make 并创建你的目标文件。
- Rebuild all 全部重建：重建整个项目。
- Clean/Rebuild all 全部清除/重建：先清除项目目录，然后重新运行 make。
- Stop Build 停止建造：取消当前程序--这是最常用的，如果你观察 make 的工作情况，仔细看你的源文件就知道。然后--啊--我忘了这件事了。。。你还必须修正你的代码。只需点击"停止"，更正你自己所发现的错误，再重新运行 Make 就是了。

但这并不是 Kdevelop 和 make 一起使用的唯一方式--对于 KDEapplications 应用，还有一些特别之处，例如为国际化创建消息文件。这些功能也被包括进来，所以你无须再担心这些事了。到现在，你已经了解了源文件，Compiler 及为什么需要 make 了。在下几节中，我们会讨论由 Kdevelop 自动创建的项目是如何在大

多数其他的使用 configure-script 脚本的 Unix 平台上被编译的。

## 3.3 设置

本节的题目很可能让你疑惑：设置？有什么必须要设置的？谁来完成呢？好，假设你已经写好了一个包含了 Makefile 的程序。然后你想要发布它，但这些编译过的二进制代码在你的系统上或是与你的系统兼容的机子上不能运行。为了支持其他平台，例如另外的 Unix 系统或是如 Alpha 机、RISC 机，你就必须重新编译这个程序。最简单的办法就是把源文件包拷贝到目标机器并再次运行 make。但要是目标机器使用的是另一种 Compiler 编译命令或者在另外某方面在建立你的二进制时遇到了问题该怎么办呢？更不要说更困难的情况了，例如你的程序和文档--例如 KDE 的安装路径，不能在一台机子上被安装到 opt/kde/，而在另一太机子上被安装到 usr/local/kde/下。在这种情况下，你就必须每次都重写 Makefile 文件，以保证你的产品的正确编译和安装。幸运的是，GNU 工具甚至还提供了比 make 还强大的工具--常用的 automake 自动创建和 autoconf 自动设置包。带 auto 的词听来总是很舒服，就好象是关于应用程序设计的东西可以又快又轻易的完成，实际正是这样。

自动创建的目的一般是从你必须为你的项目书写的文件 Makefile.am 创建一个所谓的 Makefile.in 文件。Makefile.am 文件由宏组成，它们可被翻译并可降低 make 的复杂度，所以 Makefile.am 文件可以比最终的 Makefile 更安全、更快速的编写完成。

那么，是什么最终为我创建了 Makefile 文件呢？是 autoconf 自动配置。自动配置要求项目拥有几个宏文件。那是那些由 automake 和一个叫做 configure.in 的文件创建的 Makefile.in 的文件，也包括宏。因此， Makefile.am 和.in's 都包含了宏，它决定了创建软件的方式：源文件的编译方式，哪些文件属于这个软件包，以及最终的二进制或？进制文件在创建后用的名字。在另一方面，Configure.in 则包含的宏则决定最终的配置--外壳在配置被执行的系统上将做何种检测。那可以是，例如 Compilercommand 命令最终二进制将被连接所需的库，项目所需的包含文件及其位置。

例如，你想写一个 KDE 应用。在你编写完资源代码以后，你想把你的程序发布到用户社区，而每一个用户都必须在自己的机子上编译这个二进制资源。那么你就需要写一个包含编译应用所需的宏的 Configure.in 文件。不论 Qt 库是否安装，那一个宏最终在系统上展开成一个 check，检测 Qt 头文件，KDE-libraries 和头部等等。

**总结**：为创建一个在不同的 Unix-OS 和其他机子上都可运行的 GNU-编译应用，你需要这样做：

1. 为你的应用写下代码资源
2. 为每个子目录编写 Makefile.am，包括你的项目的主要项目目录。
3. 编写放于主项目目录的 configure.in 文件,包含说明系统要求的宏。
4. 运行 automake
5. 运行 autoconf

现在主要的工作都已完成，"自动创建"建立了 Makefile.in，autoconf 启动了 configure.in 并生成一个可执行的，称为 configure 的外壳脚本。你所有接下来必须完成的事就是用.configure/执行它，脚本会运行你选中的 checks。最后会生成一个 Makefiles，允许"创建"的最终执行。它会运行所有的 Makefiles 文件，然后你就完成了。

看起来为写一个小程序，花费的人力可不少，该学的也不少，特别是如何编写正确的宏。但仅仅是你提供在几乎所有的 Unix 系统上都可运行的应用这一事实本身，迟早也是值得你的这些努力的。最后，你只为你的应用做一次这样的工作，万一你的项目的文件有所增加，你只需往宏里加入文件就可以了。

现在，Kdevelop 究竟有多支持这种类型的应用程序开发，而对程序员来说这究竟又有多复杂呢？这里有个好消息是你甚至无须知道关于宏和 scripts,的任何东西。所有细节问题都已隐藏，使用起来轻松自如。因此由 GNU 工具都以对用户有好的方式创建应用:

只需根据你的应用的需求选择，用 KappWizard 创建你的应用--那可以是一个纯 C++最终应用或某种使用 Qt 或 Qt/KDE 的 GUI 库的程序。所有工作自动完成，而且你的项目已经包含由 GDU 工具和配置脚本的自动执行创建的 Makefiles。

就是这样--你准备扩展你的项目资源，可以通过增加 classes 类，对话，翻译或文档，这些都是自动化的。只需集中精力去做开发者的真正工作，那是为最终你想建立的应用程序编制功能的。在大多数情况下，在使用 Kdevelop 时，你很可能都不会和 <Makefiles 打交道。

## 3.4 调试

以下这一、节会讲到一个开发人员广泛使用的术语：调试。它的意思是，虽然你的 Compiler 会生成最终的应用，你的应用可能无法运行或在执行时由于代码中所谓的" 臭虫而崩溃。由这种昆虫名来描述的程序错误的由来可追溯到计算

机的历史；最初引起机器崩溃的错误中，有一个并非是 malfunction--臭虫是在计算机内部而引起机器瘫痪的。因此，第一眼无法立即探测到的错误就被叫做"臭虫"。所以"debugging"调试的意思就是除去不该有的臭虫。现在，你并不需要真正猎杀它们，假设今天的计算机设计了某种外部保护，可以把臭虫排除在外。臭虫一定是在代码中被发现，大多数在完成一个程序的执行时会弹出信息框"节段错误"。GNU 还提供另一个叫做 gdb 的工具，即 GNU 调试器。这一终端程序允许监测程序的内部参数值和利用在代码中设置断点来观察程序的逐步执行。Gdb 在每次程序在执行中遇到断点时即中断程序的执行。但和大多数其他工具不同，调试器是由另一个程序控制的，该程序为它提供前端，允许轻松观测其参数值，及在代码中设置断点。

为此，你的项目的应用是由 debugging 调试的 Compiler 编译选项缺省创建的，因而在可执行文件中存储另外的数据，以允许代码中参数值和路线的定位。作为 gdb 第三方的前端，Kdevelop 使用了 KDbg，即 Kde 调试器。要调试你的程序，你只需从建立--菜单中选择"调试"，或按下由上面有幅眼镜的轮子来表示的相关工具按钮，以表明你西哪个检测程序的执行即可。

然后 Kdevelop 会打开 Tools-window 窗口并用 KDbg 启动你的应用。 KDbg 界面会出现在 Tools-window 窗口内部 ，允许你象在外部启动它一样的使用它。

总的来说，以上步骤很清楚的说明了程序员在开始写他自己的应用时，必须执行的某些步骤的必要性，并阐述了基于所有项目的一般观点。我们还解释了 Kdevelop 为一个开发者的工作所起的作用，及它是如何支持为 UNIX 编程提供简易途径这一想法的。如果想知道关于 GNU 工具的作用和目的的更多信息，请阅读随之提供的文档，一般可以通过 man 命令或在 KDE 帮助的"系统 GNU 信息内容"一节得到。

# 4. 利用 KDevelop 开发篇

本章中你会对如何使用 Kdevelop 及其内部的 dialog editor 对话框编辑器创建自己的应用得到一个总体的了解。因此，所有工具在开发过程中所起的作用都会介绍到。如果你是 C++GUI 设计和编程的新手，你可以阅读 Kdevelop 编程指南，它叙述了编程设计的基本知识，并利用项目实例对 Kdevelop 的使用做了更为详细的说明，你可以一步一步跟着做，来熟悉利用 Kdevelop 创建项目的方法。你还应该阅读 程序篇 程序篇来对 Unix 应用为什么要使用所提供的开发工具得到一个总体了解；这样在你看到关于 Makefiles 等等是什么的参考时，这会使事情变的更容易。

## 4.1 什么是 Kdevelop 应用程序？

由 Kdevelop 创建的项目让开发人员能轻易的使用 GNU 标准开发工具。和自动创建的 Makefiles 相反，它们提供的不仅仅是创建 Makefile 的一个更好的途径，还有一个更好更安全的办法，通过 autoconf 创建 configure 脚本来提供快速改写以适合不同的系统；而前者因为项目的复杂性，无论如何都必须编写。

除了一个 C++Compiler 编译器和开发库外，你的应用的发布并不要求终端用户安装其他东西，通常情况都是如此；但是你可以发布你的应用的二进制包。在任何一种情况下，你的产品的终端用户都不必要安装 Kdevelop。为分发拟订资源代码，我们建议你也包括进 Kdevelop 项目文件，这让其他开发人员能更轻易的处理你的源代码，如果他们也使用 Kdevelop 的话。对于牵涉了几个也许在不同地方工作的开发者的项目，无论如何必须保证 Makefiles 的一致性，以使你运行时不至于出错。尤其是多种语言的应用程序，翻译人员实际上并不和源代码打交道，除非它需要更正以得到翻译支持。不管怎么说，Kdevelop 通过加入他们的语言和减少他们的工作使之专注于翻译，来将其工作量减至最少，这些多时值得感谢它的。使用 Kdevelop 建立你自己的项目，你可以通过 KappWizard 应用向导开始建立新的应用。在那里，你还可以根据你所编写的程序的目标，设置项目类型。然后，你通过加入 widgets 开始开发用户界面，widgets 已经建立在库中，只需被你的应用访问，或者通过对话框编辑器自我创建。下一步就是通过改变和扩展以由 Kdevelop 制作好的那些部分，例如状态条，菜单条和工具条，扩展用户界面。由于资源代码收集在源文件中，也可以通过为已存在的量增加新值来扩充它。所给资源也可以用作增加你的特定项的准则。在实现将功能转化为方法后，

这些方法可能是由 dialog editor 生成为空，可能已包含标准用户操作的代码。你应该通过简单编辑所提供的 SGMLindex file 索引文件描述你的应用程序的性能以扩充用户手册。最后一步即是生成翻译文件和把项目分发给翻译人员来完成剩下的工作，以增强项目的国际化。

但即使对于非 GUI 应用设计人员，IDE 也提供一个简便的方法来创建新项目；编辑器和类编辑器加上 Kdevelop 强大的项目管理器，可以将开发时间缩短至功能的 implementation 实现所要求的最短值。

## 4.2 关于 KdevelopTools 工具的简要描述

利用 Kdevelop 你以选择一个软件包，与其他工具一起使用，它会提供给你一个完美的，伸缩度高的环境来在 Unix 系统下进行面向对象的应用程序开发。总的说来，Kdevelop 由编程环境，dialog editor 对话框编辑器组成；除此之外，其他所需的对设计应用有帮助的程序都以某种方式安置于内。

### 4.2.1 编程环境和对话框编辑器

#### 4.2.1.1 编程环境

编程环境是 Kdevelop 为创建和支持项目所提供的用户界面，无疑它还使用了很多其他工具，但简化了其使用。它内置的编辑器和 Helpbrowser 帮助浏览器，以及类浏览器减少了开发中通常是由程序员承担的工作量。使用简便使得在 Unix 系统下的编程本身成为一件乐事，很可能会导致发布周期缩短，开发质量更高。从此，你可以通过所有细节管理你的项目，通观全局管理好你的 classes 类，而无须为工作启动另一个控制台，而所有的用户界面的集成也使你只需在 X 下在一个窗口内工作，而不必监视桌面上铺满的众多窗口。

#### 4.2.1.2 对话框编辑器

Kdevelop 内置的对话框编辑器提供利用 Qt 和 KDElibraries/设计用户界面的简便途径。它利用鼠标做几乎所有的事，这一简便方法允许为应用和应用直接转换为 C++代码快速创建对话和主视，这是直接被加入项目的。dialog editor 已被内置于余下的用户界面，你的可用命令和工作在编程模式下一样。这就允许了例如对话的创建，output 输出代码的生成和它在程序中利用 make-commands 的可用

性直接测试，你仍然可以控制 Compiler output 编译器输出，而无须重新切换回编程界面。要想得到更详细的信息，请参见 对话框编辑器。

### 4.2.1.3KAppWizard 及类工具

Kappwizard 及类工具为项目生成提供条件，而自动代码延展旨在让应用程序的生成尽可能的简单和安全，它们也为 Qt Kappwizard 编程及类工具初学者来亲自实际工作，并深入应用程序的细节，这样可以快速的得到结果。

### 4.2.1.4KAppWizard

Kappwizard 旨在创建功能完备的应用项目，这些项目利用 GNU 标准工具随时可运行，而对 GUI-programs,则是利用了 KDE 和 Qt 库。通过仅仅指定所要求的最少信息，用户可以在数秒钟内开始一个新的项目的工作。访问 KappWizard 是利用 Kdevelop 为用 Kdevelop 创建一个新的应用所需采取的第一步。

Kappwizard 提供给你一个包含了运行应用所需的所有 C++源代码的所谓的应用框架。在你指定项目的名称，设置了所需功能，你的新项目就可以生成，你就有了一个已包含很多功能（对 KDE 或 Qt 库）的完整应用。甚至还无须亲自打出一行 C++代码。当指定程序生成起提供的一整套的选项，你的应用程序就已经包含了：

- 一个 SDI 界面工具（每个应用窗口可对一个文档操作），基于文档--浏览--控制者模式。
- 为打开和保存文件设置的菜单和状态条，和 priting 打印对话框一样。
- 一个工具条，状态条，以提供完整功能。
- 一个帮助系统，包括一个基于 SGML 的用户手册和状态条帮助。
- 为项目当前状态设置的一个完整的基于 HTML 的 API(应用编程界面)文档。
- 通过 make 建造的安装路径。

KAppWizard 通过项目菜单的"新建"项可用。

### 4.2.1.5 类工具

类工具这一术语描述了开发者可于项目的各种功能，它不仅通过面向对象的语言，还有以对象面对的方式使用代码资源。使用 classes 可以以很灵活的方式扩充项

目，容纳感用户将注意力集中于项目包含的类上。Kdevelop 提供的工具是类生成时的类浏览器。每个都提供了许多自动功能。

#### 4.2.1.6 类生成器生成新类

在用 Kappwizard 生成一个项目后，你作为开发者的工作会增加一套 classes 类，以匹配你的应用程序的设计。不使用 Kdevelop 的经典程序是创建一个新的头部和源文件，手动将之加至相关 Makefile.am 并开始编写 classes,declaration 和 implementation。为简化整个过程，类生成器让你仅需通过以其名字指定新的类。被继承的类，继承属性及更多的信息，比如项目属性的类文档来完成它。另外，你还可以改变生成器预先设置的文件名，及类是否缺省地设置为 Qt-signal/slot 可用。

余下的工作会自动完成，你无须关心 Makefile.am，文件和类本身。一个新的"建立"已足够在编译过程中包含新类。类浏览器会在增加类后自动更新，因而 declaration 和 implementation 立即可用，你也可以开始着手于类的扩展工作了。类生成器可在项目菜单中，通过"New Class"项进入来激活。

#### 4.2.1.7 类浏览器

类浏览器在编辑器窗口左边以数的形式展示你的项目所有类型的对象和功能。类语法分析器为方法和 classes 扫描所有的文件，并让它们通过符号可用。除此以外，文本菜单提供一个专业化的功能，通过专注于 classes 和函数着手于资源代码。

选择一个将导致包含 declaration 的文件被打开；在方法和函数上它会给你看 implementation。要想得到完整的描述，请参见 类浏览器。

## 4.3 开发过程

用 Kdevwelop 开发的应用一般可以分为两个主要的步骤：首先，你必须利用 Kappwizard 生成一个程序轮廓，然后剩下的开发工作必须利用 Kdevelop 的特色来完成。它只要求你作为程序员的工作能力。为描述开发过程，我们假设你希望建立一个 Qt/KDE 应用，它说明了需要使用 Kdevelop 的大多数特色来着手于这些项目类型，因此你有一个一般的做法。

### 4.3.1 建立程序框架

开始创建你的应用，你一般需要访问 Kappwizard；输入项目名并生成程序，通过这一步骤，应用向导会自动为应用创建项目文件并将 Kdevelop 区域插入在随后展开应用时会用到的 Makefile.am's,项目文件是你在以后打开项目时必须装载的文件。

你通过项目菜单进入"新建"项，访问 Kappwizard。当向导出现时，你还必须在第一页指定项目类型，选择"下一页"会将你带到下一页，你必须在此插入项目名，版本，目录和你的个人选项，第三页会提供更多个人选项，第四页和第五页允许根据你的选择自动插入头部和 implementation 编辑头文件。如果你已经有了你想要使用的头文件，你也可以选择那些文件。最后一页包含了一个 output 窗口和一个错误信息窗口。当选择"生成" 时，你的应用即被生成。所执行的操作在 output 窗口可见。如果"准备"出现在 output 窗口，Kappwizard 已经完成，你可以通过选择"退出"返回编程环境。为建立二进制文件，你还可以通过从创建菜单中选择"make 生成"，或选择工具条上的相关按钮。你还可以通过从创建菜单中选择"执行"来测试已有功能。

### 4.3.2 开发一个应用

本节将描述使用 Kdevelop 和 dialog editor-开发应用程序的开发阶段--所有的工作都可在 IDE 中完成。

开发步骤一般说来涉及编辑源文件，编译项目代码及 Linker 连接所有对象文件至最终二进制的连接过程。妨碍编译连接的错误 errors 或是通过 debugging 调试会话可以查找到的臭虫都应该找出。最后，文档必须要扩展，翻译必须要加注。但以上各个步骤都是可以互相混杂，要想描述如何建立你自己的应用程序的一般方法并不是那么容易。典型的情况是，通常的程序先创建项目所有的可视部件，例如，主视，和设置路径的对话框，设置选项的对话框；在生成源代码，实现所有需要的连接，例如：改变菜单条和工具条，为新菜单词条加入状态条帮助，并用显示你的对象给新的成员填充功能。然后你就可以编译你的应用，测试其性能，修正 errors，再重新检测。后面几节会说明这些工作是如何完成的；为建立你特别的应用，你可以只选择另一种方式--Kdevelop 给你足够的灵活度来决定什么时候做什么。

### 4.3.3 用户界面

一个项目的用户界面实际上是用户所见的部分及它和程序的桥梁。因此，一个程序一般都含有菜单条，工具条和状态条，就象有主视的一样（例如编辑器的文本框键入）， Kappwizard 创建的应用已经包含了一个复杂的功能--框架结构会包含标准元素，例如条和按钮。当你开发你的应用时，你必须扩展给定的结构才来让程序有你想要提供的交互性。这一工作的步骤之一就是对话框设计，例如，改变画图应用中的笔刷宽度值。这可由 Kdevelop 内部的对话框编辑器 dialog editor 来轻易完成。现在，你如何创造那些美观的对话框呢？由于 Qt-library 是 GUI 工具箱，它提供了一个所谓的"widgets"基本套件；例如，表示纯文本的标签，文本输入的行编辑，导航按钮及选择命令。用户界面中除了这些"低水平"的部件，还提供标准的对话，随时可用，例如打印对话框 printing 。KDE 库 KDE-libraries 是基于 Qt-库的，它包括某些对话的替代及可用于你的应用的其他 widgets 。这样，当你开始设计你的用户的应用操作时，你应该让你自己尽量的熟悉随 Qt 和 KDE 提供的 widgets。Qt 的在线文档为大多数 widgets 提供了快照，因此你应该先去看看。然后我们建议你仔细看看 KDE- UI 库, 它提供了其他一些 widgets。象 KHTML 库一样的库包含了很专业化的 widgets ，很多程序都在使用。它只是简化了应用的设计，例如 Qt 提供的预先定义的对话。对于你自己的对话，这些较低级的 widgets 正是你所需要的。一个对话仅仅由不同的 widgets 结合在一起组成了一个界面，它在技术上是由 C++代码编程于一个从 QWidget 中派生出的一个类，或一个更专业化的继承了 QWidget 的对话类。因此使用 widget 的程序需要一个 widget 类的对象。--这几乎是你要理解对话及视图是如何使用的所有东西。

现在 Kdevelop 的对话框编辑器 dialog editor 提供一个很好的方法来简化你的对话的建立过程，它可以可视化得创建对话和用户界面，你无须再将对话和纯 C++ 代码结合再一起。 --这是创建界面的较为复杂的方法。通过选择视图菜单中的相关条目可切换至对话框编辑器 dialog editor (或通过工具条按钮)。你可以看见 Kdevelop 的界面改变了，但是，菜单条和工具条仍然看来十分熟悉。这使得在两种工作模式间前后切换变得十分容易，你很快会习惯于它们的。然后你可由你所愿建立你的视图和对话框。并设置所有视图包含项目的可用设置。当你完成后，从建立菜单中选择"生成源"；对话框编辑器 dialog editor 及项目管理器会负责剩下的事情。你可以通过选择"生成"或"执行"来测试是否一切正常，这会建立你的包含了你的新资源的应用。但是不要期望对话框已经可用--这通常是一个开发者工作完成 implementation 的一般方式。但不要担心--这也并非那么困难。要想得到关于创建对话的更多信息，请参见 对话框编辑器, 实例及指导可参见

## 4.3.4 捆绑新元素

在你创建了你的用户界面并生成了你的资源后，你可以准备让你的应用通过对象来使用他们。如上所述，通常一个基于 GUI 的程序会包含菜单条和主视；另外主视可由控制条操作,它是主应用类及其连接的文档对象的一部分。按照面向对象的设计，你可以将给出的结构描述为"文档--视图--控制器"模式。这种模式通过介绍对象在程序中的作用描述了对象在应用中的基本功能。控制器代表协调主应用的一类，它通过菜单条及工具条，状态条提供用户交互功能。文档类的任务是代表用户使用的文档。因此，文档类应该可以完成所有的比如装载文件和重新保存文件的操作。现在视图位于应用窗口中央，为用户以可视化方式显示文档的一部分，并提供所有功能使用户可改动数据。由于控制条和工具条已出现，你的工作将是创建主视，以及用户可以用于改变任何设置或使用另外功能的其他对话。

要建立你的主视，为你的应用生成的源代码已经以<YourApplication>形式包含了一个类，继承 QWidget 类的视图(这是操纵 Qt 中可视化界面的最小类，因此也在 KDE 中)。 总的来说,展开给定的来有三种不同的方法:

- 去掉文档--视图结构并使用一个预先定义的已包含了很多功能的"大"widgets，只需去掉视图类并由另一类代替。
- 改变视图类的继承，改变至，例如，QmultiLineEdit，这样你的应用会变成一个编辑器。
- 如果你的主视窗口要包含若干独立部件，你就需要用 dialog editor 创建你需要的 widgets 及其 classes，并在连接后可生成主视的视图类生成器中创建该 classes 的对象。

对对话来说，情况就有些不同了。通常的做法是从菜单条或工具条上按钮进入，访问对话框。在你创建了你的对话并生成资源代码后，类就可用于创建作为对话的对象了。所以你先看一下，在菜单条找出适当的位置来加入用户选择后可打开对话的入口。如果给出的菜单与你的要求不符，你可以创建一个象其他菜单一样的新的弹出菜单，插入你的项目及当你的菜单被选中后你想要访问的地方。通过创建对话类实例实现 slot，并用基础类提供的成员函数 member functions 访问对话框。而且你还必须为菜单项指定 ID。应用程序框架把已给出的 ID 收集于文件 resource.h 中，因此你只需加入 ID 并给它一个新编号。这样你就完成了--另外你还可以创建另外的工具条按钮并加入状态帮助信息。现在你们的应用给用

户提供了可视的新功能。接着你必须加入你的对话将用到的操纵任何参数值的方法的实现 implementation。最后，访问你的生成工具或"执行"，改写过的文件会被重新编译；这样你改动的结果可以立即测试。

### 4.3.5 完整开发过程

以上我们已经讨论了关于如何利用 KDE 开始建立新的应用及如何扩展用户界面的一般方法。现在，这些是 IDE 帮助你的标准步骤，但 KDE 却不仅仅为你提供创建应用及其可视化部件的工具。下面我们将对它对推动应用开发提供的功能做一个简要描述。

### 4.3.6 源代码管理

Kdevelop IDE 提供给编程人员很多的方法使他们在最短的时间内完成其目标。如以上所述，KappWizard 和 dialogeditor 缩短了你手工要达成同样结果一般所需的时间。但这并不包括一个程序员通常必须做的工作：为应用的完成努力工作，以使终端用户能正确的执行。那么，为什么 KDE 是你希望用做编程环境的 IDE，包括它的即使是非 GUI 应用的创建作用？

无疑，IDE 总体的管理着你的项目；那就是说，你无须小心对变化的保存，Makefile 的生成，等等--这提供了整个项目的管理，KDE 会在任何一个 C++应用开发中不遗余力的为你服务。你可以很容易的理解，把管理的任务从程序员手中接过后，他可以更专注于源代码的编写工作了。在通常遍布于项目中许多不同的文件，因此你可以分离一定的文件。但这意味着仍然要努力工作--作为开发人员，你仍然不得不创建这些文件和编写这些标准内容，例如包含了数据，作者名的头部文件，及例如代码的执照术语。另外，这要求你必须记得你的函数，及类 declarations 和 implementation 在何处。因此，KDE 包含了类工具 --各种各样的操作，他允许快速操作和将开发者的注意力从文件转移至对象 classes，结构和方法。类生成器让你可以轻易创建一个包含继承，attributes，和文档的新的类。对于面向对象的工作，类浏览器把你带到你的对象的位置；而代码实际在哪个位置已不再有什么关系了。浏览器自动扫描所有资源并在有增改时自我重建以跟上你的工作，使你可以直接得到新的代码。通过上下文菜单,类浏览器会提供更多的功能，例如，将你带到实现 implementation 或成员函数 member functions 的声明 declaration。然后，成员的增加可以通过对话来完成--不必寻找文件和你要加入的项的位置。最后，你可以通过类工具对话得到你的项目的类 classes 的更为

专业的视图，它会以树的形式显示类 classes 及其对象的作用，内容和继承。要想得到更详细的信息，请参见 类浏览器

### 4.3.7 建立并执行你的应用

Kdevelop IDE 是特别设计来为你完成你必须定期执行的所有那些步骤，例如建立和执行你的程序及在源代码中定位 errors。

你可以开始你的建立过程，通过：

- 单击工具条上的"建立"或"全部重建"符号
- 或从"建立"菜单中选择"建立"或"全部重建"

要执行你的应用，请选择

- 工具条中的"执行"或"编译"符号(由你的程序启动 KDbg )
- 建立菜单中的相关菜单项
- 或者通过"由 Arguments 执行"来用其他 arguments 启动你的应用

要想得到关于建立过程的更多信息，请参见 项目篇。

### 4.3.8 寻找程序错误

由于一般 errors 错误会在创建过程（由 Compiler 编译器监控 ，并且由于它们源于代码的句法错误，一般被称作句法 errors 错误。）或应用的执行过程中发生，它们必须由程序员找到并除去。 要定位 errors 错误,开发者需要得到引起错误发生的确切信息。 如上所述，Compiler 编译器本身可以检测句法 errors 错误,导致可执行文件无法建立(这在 Linker 连接器检测到"unresolved 信号时也有可能发生 --参见连接器选项 连接选项). 它会尽可能详细的对错误作出描述，这样错误就可以找到并去除。在创建过程,你可以看见 output 输出窗口弹出，告诉你你的 make 工具和编译器 Compiler 要说的话。万一遇到了错误 errors 或者警告 warnings, 只要在错误行按动鼠标按钮，编辑器就会打开该文件并将光标移到错误行。这也可以通过"视图"菜单中的"下一个错误"和"上一个错误"条目或通过相关键盘快捷方式 shortcuts 来完成。你会发现这十分好用，为你到达错误节省了不少时间,这样排除状况只要求你作为程序员的相关知识。实时错误是在执行时出现的，大多数情况下会导致节段错误，有时很难发现。因此，你可以让编译器

Compiler 把信息加入二进制文件以监测源代码的执行。调试器则是允许你通过启动应用并在代码中为需中断执行的地方设置断点来完成它的另一类程序，因此你可以严格控制其实现 implementations 而且通过这样至少可以检测引起错误发生的行。找出真正的引发原因则是另一项任务；它要依靠程序员对它的定位。**gdb** 是 GNU 编译器 Compiler 提供的调试器，象 *ddd* 或 *KDbg* 一样的程序是允许更方便使用的前端。因此 KDevelop 使用 KDbg 并让你可以通过"建立"菜单中的"调试"命令或工具条中的"调试"标志来激活调试 debugging 程序。要想知道关于调试 debugging 应用的更多信息，请参见 项目篇 及 KDevelop 编程手册 。

## 4.4 其他信息

本章的话题将更详细的在 KDevelop IDE 提供的文档和其他文档中叙述以下内容：

- KDevelop 编程手册 ,完整的介绍了使用 Qt 和 KDE 库进行 GUI 的应用设计和编程,
- 这本手册,从 概要一节到 项目一节，描述了 KDevelop 的所有可用函数,
- 这本手册,在 对话框编辑器一节，叙述了对话编辑器 dialog editor 对于创建你自己的 widgets 所起的作用,
- 在线 *Online*-参考文档到 Qt 库 Qt-library, 包括了使用 Qt 的 GUI-toolkit 工具箱及类参考和快照创建其包含的最重要的可使用的 widgets 的实例，
- *KDE-Library Class-Reference*，由 KDevelop 从 KDE-库资源自动生成,包含了所有类 classes 和 widgets 的描述及其使用的代码实例，
- 在 Internet 上，参见：
    - http://www.troll.no 以得到关于 Qt 和其他第三方 widgets 的信息,
    - http://www.kde.org 以得到关于 KDE 项目和开发者指导,
    - http://developer.kde.org 以得到其他 KDE 应用开发的的参考.
    - KDevelop 的主页 http://www.kdevelop.org

# 5. 概要

看看 KDevelop,其用户界面可由其逻辑部件来描述: 主视窗,树浏览及输出 output 窗口,由菜单条，工具条和状态条所包围。本节将描述界面中每个部件的目标,先从窗口讲起，再说控制条及其提供的函数。

## 5.1 主视窗

主视窗由四个状态条组成，左边两个用做编辑。先说说头部 Header 中为头部和其他任何文本文件，例如文档 SGML, 所设的资源窗口，然后是为源代码所设的 C/C++窗口。它们之后是表示成 HTML 格式的文档窗口。最后是为内部编程，例如以内置于 IDE 的 KIconEdit 和 KDbg 而设的 Tools 工具窗口，已注册的任何工具都可以通过 Tools 菜单来使用；而第三方程序的注册可以通过 configuration dialog 设置对话框 (参见 一般设置)来轻易完成。要想得到关于编辑器窗口性能的描述,请参见"使用编辑器"篇 使用编辑器 ,对于 Helpbrowser,则需参见"使用文档浏览器"一节 使用文档浏览器.

## 5.2 类浏览器和文件浏览器

### 5.2.1 类浏览器 Class Viewer

类浏览器 Class Viewer (CV)展示你的项目的各个类 classes 及全局函数和变量 variables。打开它的树会给你展示所有的函数成员 member functions 和带 attributes 属性标志的成员(私有的,有保护的和公有的,signals and slots),因此你可以看见可视化的成员属性，而无须切换到头文件。 选择一个类名会打开相关的包含该类的头文件并将光标设置到类的声明 declaration 的起始处。选择成员函数 member functions 会打开 implementation 文件并将光标设置到函数头部。对于元素，KDevelop 会找寻头文件并将光标设置于元素声明所在行。类浏览器 classviewer 还提供弹出菜单，包括更多专业的选项，请参见 类浏览器查询更多详细信息。

### 5.2.2 逻辑文件浏览器

逻辑文件浏览器(LFV)允许通过各组中的过滤器将文件分类。独立的小组可以"项目 "菜单中的属性项或在树的条目上右击来加入。这会允许对你想快速得到的文件提供更专业化的搜寻，只首先显示包含项目的文件。文件会在选中后根据它们的 Mime-类型打开。你会喜欢 LFV 的，例如用它完成选择 pixmaps 一类的事--这会在 Tools-工具窗口中启动 KIconEdit(如果已在系统上安装)并打开所选图片。

### 5.2.3 实际文件浏览器

实际文件浏览器(RFV)将所有文件表示于项目目录树,因此你可以编辑非-project files 项目文件或被 LFV 隐藏的文件，例如 configure.in 和 Makefiles。弹出示菜单也提供例如增加或去除当前项目的文件的功能。

### 5.2.4 文档树

文档树(DOC)展示所有可用的基于 HTML 的被设置为书的文档。选择其中一本书会在浏览器窗口中打开其第一页。而且,弹出菜单会提供另外的基于 HTML 的文档包的个人设置。

## 5.3 输出窗口

输出窗口也被分割为一个信息窗，一个标准输入/输出窗口和一个标准错误指示窗口，属性如下：

- 信息窗口:展示所有编译器 Compiler 的输出 output。点击错误信息会改变编辑 widget，将光标设置到编译器 Compiler 发现错误的行。
- 标准输出窗口: 展示将信息发至计算机标准输出 output 的基于终端的应用。 注意，终端应用现在将在一个外部控制台的窗口中启动。
- 标准出错窗口: 展示你的程序产生的所有错误信息。这对于测试来说很有用。输出窗口在编程时被设计为每次外部程序被访问时都会出现，例如 make 或一个终端应用。

## 5.4 菜单条命令

### 5.4.1 文件管理和打印

本节将描述 Kdevelop 为文件提供的功能，可通过菜单条中的文件菜单或工具条中的相关按钮进入： 文件菜单

- **新建 Ctrl+N** 打开新文件 "New File" 对话, 允许创建新文件。 文件可用不同的 templates 创建，文件名及创建文件的路径也须给出。
- **打开 Ctrl+O** 显示打开文件对话，让你选择一个要打开的文件。
- **关闭 Ctrl+W** 关闭在顶部编辑窗口中的文件。
- **保存 Ctrl+S** 保存顶部窗口中打开的文件。如果文件还未保存, "另存为" 对话会打开，让你选择文件将被保存的的路径和文件名。
- **另存为...** 打开"另存为"对话，让你以新文件名保存当前文件。
- **全部保存** 保存所有改动过的文件。
- **打印... Ctrl+P** 打开"打印"对话，你可在此使用 a2ps，enscript 或 lpr 设置打印 printing 选项。
- **退出 Ctrl+Q** 退出 Kdevelop。 如果有文件被改动，你会被询问是否愿意保存这些文件。

### 5.4.2 编辑文件

"编辑"菜单在此会描述提供文件编辑的"编辑"菜单和工具条中的相关按钮。编辑功能也可通过编辑器 editor 中的文本菜单进入。

- **撤消 Ctrl+Z** 取消上次编辑操作。
- **重做 Ctrl+Y** 让你重做上一次取消的操作。
- **剪切 Ctrl+X** 剪切所选内容并将它拷贝到系统剪切板。
- **拷贝 Ctrl+C** 拷贝所选内容到系统剪切板。这对文档浏览器中选择的内容同样有用。
- **粘贴 Ctrl+V** 将系统剪切板中内容插入光标当前位置。
- **缩进 Ctrl+I** 将选取内容右移。
- **取消缩进 Ctrl+U** 将选取内容左移。
- **插入文件...Ctrl+Insert** 让你选择一个文件并将其内容插入光标当前位置。
- **搜寻... Ctrl+F** 打开搜寻对话，寻找当前文档中的某个语句。如果要在若干文件中找寻,应使用"在文件中找寻。。。"。

- **重复搜寻 F3** 重复上次对某语句的搜寻。这对同一页中找到多处匹配结果的文档中的搜寻也有用。 选择"重复搜寻"或按下 F3，下一个搜寻结果会被标注显示。
- **替换... Ctrl+R** 打开"寻找并替换"对话，允许搜寻某语句并用新的语句替换找到的文本。
- **在多个文件中搜寻... Ctrl+Alt+F** 显示"在多个文件中搜寻"对话, 在整个目录上用 wildcards 处理 grep 。搜寻结果会将文件名，行和语句列表显示。选择一项将打开该文件并将光标设置到搜寻结果 search result 所在行。
- **全选 Ctrl+A** 选择当前在顶部编辑窗口中打开的文件的全部文本。
- **取消全选** 取消选择当前文件的全部文本。这在有多行选择时常常用到，这样你就无须逐行取消选择了。
- **翻转选区** 翻转选择的内容,这意味着选中文本变为未选中，而未选中文本变为选中。

## 5.4.3 视图设置

"视图"菜单视图菜单包括窗口的开关功能及显示/隐藏工具和状态条、在代码中跳过错误 errors 的命令 。

- **跳到...行 Ctrl+G** 打开"跳到。。。行"对话，让你插入在现行文件中想要显示的行号。 最后一次输入的行号会被记下并标注, 因此你可以再次跳到该行，或输入你想浏览的新行号。
- **下一个错误 F4** 跳到 Kdevelop 从 output 检测到的下一个错误。Make 或其他工具的输出 output 信息会为你作出描述，帮你找到问题所在,这样你就可以更正错误了。
- **前一个错误 Shift+F4** 跳至前一个错误。
- **对话框编辑器 Ctrl+D** 切换至对话框编辑器 dialog editor 。
- **树的浏览 Ctrl+T** 在主视窗的左边显示/隐藏树窗口，包括类浏览器, LFV,RFV 和 DOC-tree.
- **输出视图 Ctrl+B** 在主视窗底部显示/隐藏输出窗口。
- **工具条 Toolbar** 显示/隐藏工具条。
- **浏览器工具条 Toolbar** 显示/隐藏浏览器工具条, 包括浏览器的返回，前进，搜寻按钮。

- **刷新** 重新扫描所有文件以重建类浏览器。扫描过程将在状态条的进程显示中可见。

### 5.4.4 创建和支持项目

本节将描述"项目"菜单中可用的功能，及项目的创建和支持。

- **新建...** 启动 Kdevelop 应用向导 KAppWizard ，允许你通过选择应用类型，名称，版本和其他选项，创建一个新的项目。
- **打开** 显示打开项目对话, 你可在此选择要打开的 KDevelop 项目文件。选择后项目将被装载。
- **打开近期项目...** 包括一个含有最近 5 个被打开文件的子菜单。使用"近期项目 "菜单你可以更容易的打开一个项目。
- **关闭**关闭当前项目。通常在你退出 Kdevelop，激活应用向导 KAppWizard 或打开另一个项目时这都是自动完成的。
- **新类 New Class...**启动类生成器，创建将被加入当前项目的新的类。类生成器让你指定类名, 继承，及新类的文件名。
- **加入已有文件** 打开选择对话，你可在此选择你想要加入当前项目的文件及它们将被拷贝的路径。如果你选择了项目中当前未包含的目标地址,例如，创建一个新的子目录， Kdevelop 会拷贝文件，将其加入新的子目录，并上成一个新的 Makefile.am。在拷贝程序以后,你的项目会由 automake 和 autoconf 重建以把新的子目录包含到 make 程序。
- **加入新的翻译文件** 打开一个语言选择对话，让你选择将加入项目的翻译文件的语种。通常由翻译人员使用，这样他们可以轻易的将他们的语言加入项目，翻译时也无须顾虑 Makefile.am's 了。
- **文件属性 Shift+F7** 打开文件属性对话，显示 LFV 的拷贝及项目文件选项。你的文件属性必须在此设定，例如安装路径和文件类型。
- **消息和合并** 旨在为你的项目创建消息文件。消息文件包含了为项目的国际化建立在资源文件中的所有字符串。它将由翻译人员用于为其目标语言创建 .po 文件。
- **生成 API-Doc** 访问你头文件上的 KDoc 并用你的项目的类 classes 的文档生成一个 HTML 输出 output。

- **生成用户手册** 在你的手册的 SGML file 文件上运行 KSgml2Html，生成一个 HTML 拥护手册。如果 KSgml2Html 尚未安装,将使用 sgml2html 来完成。
- **生成发布版:**
    - o **tar.gz** 在项目目录中生成你当前项目的发布文件，它包含了你的项目的各类资源，将发布到终端用户手中。终端用户必须 tar zxvf yourproject.tar.gz 来将它解压到一个资源目录，并使用标准命令./configure, make and make install.
- **选项 F7** 打开项目选项对话，让你改变你的项目的各种设置。这可能是一个新的版本号或是编译器选项 compiler options ，选择化的发布设置为 -02。

## 5.4.5 建设项目

"建设"菜单本节描述项目菜单，它包括了利用 make 要完成的所有操作，或例如，重建项目在线帮助或 API documentation 的操作。

- **编译文件... Ctrl+F8** 已可用, 如果顶部编辑窗口是 C++ 窗口。它只编译当前的 implementation 文件，万一你怀疑有 errors，可让你节省时间。
- **生成 F8** 激活你的项目的 make-command 并建立目标。
- **全部重建** 重建你的项目的所有对象文件及目标文件。
- **全部清除，重建** 从所有由 make 创建的文件中清除项目目录并重建目标。
- **停止建立 F10** 终止当前程序。
- **执行 F9** 在用 make 建好程序后执行你的目标文件。
- **以 Arguments 执行 Alt+F9** 用 arguments 执行你的目标。首先，一个对话出现，让你指定执行 arguments (可为下次操作保留于项目中), 然后你的应用使用键入的命令行。注意，你的应用程序是从项目目录中直接启动。
- **调试...** 在工具窗口 Tools-Window 中打开 KDbg 以调试你的应用。这样，KDbg 自动打开包含 main() 函数的文件并执行你的应用。
- **Distclean** 去除由项目创建的所有文件，例如对象文件等。 Distclean 必须在发布你的项目以前完成, 这样发布文件就不会包含任何平台特有的文件，比如由你的编译器 Compiler 生成的那些文件。
- **Autoconf and automake** 在文件 Makefile.dist 上访问 Make, 定位于你的主项目目录中。 Makefile.dist 包含了创建你的项目的 automake, autoconf

等的所有命令。 如果你手动增加文件或自己更改宏 ，你应该在其后运行 Autoconf，接着进行设置以重建所有的 Makefiles.

- **Configure** 执行 由 autoconf 生成的设置脚本。如果此命令无法执行，请运行 Autoconf 再重新设置。

## 5.4.6 访问工具 Tools

工具菜单 Tools-menu 缺省的包括了以下程序（如已安装）的入口：KDbg, KIconEdit 和 KTranslator。这些是由 KDevelop 安装程序来检测并以给定顺序插入菜单。 激活一个工具会打开工具窗口"Tools"-window 并在此窗口内启动选中程序。 工具 Tools 菜单可在选项菜单里由 Tools-entry 编辑；参见 配置工具菜单.

## 5.4.7 改变 Kdevelop 设置

选项菜单 选项菜单包括激活设置对话 configuration dialogs 所有条目，可用于更改 Kdevelop 的缺省设置。关于 editor 或 printing 的主要设置有它们自己的条目； Kdevelop 操作的总体设置可用 KDevelop Setup 对话完成。

- 编辑器 **Editor...** 允许进行编辑器的行为设置，例如断字，选择等。
- 编辑器颜色 **Editor Colors...** 你可在此进行编辑器 editor 的颜色设置，例如背景色。
- 编辑器缺省 **Editor Defaults...** 此处可设置缺省显示，例如编辑器 editor 的字体和字体大小。
- 句法-高亮显示 **Syntax-Highlighting** 这个对话框可让你设置几种要高亮显示程序语言的字体和颜色,包括 HTML.
- 文档浏览器 **Documentation Browser** 在这个 tab-对话中，你可以为内部浏览器 Helpbrowser 设置字体,大小和颜色.
- 设置打印机 **Configure Printer...** 包括打印机设置对话 configuration dialogs 的项目,根据使用的打印程序 printing program, a2ps 或 enscript.
    - **a2ps** 用 a2ps 进行打印机使用设置. 参见打印以获取更多信息.
    - **enscript** 用 enscript 对打印机进行使用设置. 参见打印以获取更多信息.
- 工具 **Tools...** 打开工具 Tools 设置对话. 你可在此通过增加或去除将在工具窗口 Tools-window 中启动的程序来设置工具"Tools"菜单.

- Kdevelop 设置 **KDevelop Setup** 打开 KDevelop 设置对话. 第一个 tabulator 可进行一般设置, 接下来是按键设置及文档设置. 文档设置还包括一套新的库文档 HTML-library documentation 的生成及帮助浏览器 Helpbrowser 的搜寻索引 search index 的重建.

## 5.4.8 窗口菜单

窗口菜单 窗口菜单包含一个当前所有打开的文件的列表. 这允许快速切换到另一个你正在进行操作的文件.

## 5.4.9 管理书签

书签菜单 书签菜单旨在加入和去除你想在当前编辑的文件中设置的书签 bookmarks .由于 Kdevelop 使用了两个编辑窗口, 每一个都独自设置其书签 bookmarks .

- 设为书签 **Set Bookmark** 打开一个包含了 9 个可设置的书签 bookmarks 的文本菜单。允许在书签 bookmarks 菜单中通过逻辑关系给某条目设置书签项。
- 加入书签 **Add Bookmark Ctrl+Alt+A** 将当前光标所在位置的行作为一个书签加入书签菜单 bookmarks。注意这可能覆写一个由"设为书签"选项设置的书签。如果浏览器已打开, 书签会被加入浏览器窗口的书签菜单 bookmarks menu。
- 清除书签 **Clear Bookmarks Ctrl+Alt+C** 清除顶部编辑窗口或浏览器中的书签条目,例如，如果头部窗口可见，你选择清除书签, 头部窗口中的条目将被删除。
- 头部窗口 **Header-Window** 包含头部窗口的书签列表。选择一个书签会将光标设置于选中的书签 bookmarks'所在行。注意，书签 bookmarks 只分配给他们被设置连接的文件，因此如果你改变为另一个文件，书签 bookmarks 并未被删除，但是选中书签也后不会切换至他们被分配的文件了。
- C/C++-窗口 **C/C++-Window** 包括 C/C++-Window 的书签列表。界面和头部窗口相同。
- 浏览器窗口 **Browser Window** 包括浏览器书签 bookmarks。选择一个书签会由选中的书签打开浏览器。

## 5.4.10 在线帮助 Online Help

帮助菜单 帮助菜单包括在帮助浏览器 Helpbrowser 中用于导航的及最近使用过的库和在线文档的项目。要进入另外的在线文档，可在树视图中使用 DOC-tree ，如果自动切换可用，在切换至文档浏览器窗口时树视图将自动打开。

- 后退 **Back Alt+ Left Arrow** 打开当前页的前一页。
- 前进 **Forward Alt+ Right Arrow** 打开浏览器历史记录中的下一个页面，在执行过后退操作后可用。
- 搜寻标注文本 **Search Marked Text F2** 扫描搜寻索引，找寻编辑窗口或浏览器窗口中当前标注的文本。搜寻后显示一搜寻结果页面，让你选择要切换的帮助页面。选择某页后，浏览器会以高亮显示找到的词条。如果一页上有多个匹配的词条，按下 F3 键，会显示同一页上的下一个搜寻结果 search result 。
- 查找。。。帮助 **Search for Help on...**打开查找帮助...对话，让你查找你希望得到帮助的特定语句。
- 用户手册 **User Manual F1** 打开用户手册 KDevelop 索引页以进入用户手册。
- 编程手册 **Programming Handbook** 打开编程手册 KDevelop 编程手册索引页 以进入编程手册。
- **每日一贴 Tip of the Day** 打开每日一贴对话，告知你 Kdevelop 的特色。
- **KDevelop 的主页 KDevelop Homepage** 如果 Internet 可用，将在浏览器窗口中打开 Kdevelop 的主页。
- **Bug 报告 Bug Report...** 打开 KDevelop Bug 报告对话，你可在此通过 email 将一个 bug 报告直接发送给 Kdevelop 小组。参见 Bug 报告
- **C/C++ 参考 Reference** 显示语言参考索引页。如果参考尚未安装,将显示一个错误页面告诉你如何得到并正确安装参考。
- Qt 库 **Qt-Library** 切换至你的 Qt 库的拷贝提供的 Qt 库文档的索引页。
- KDE-Core-库 **KDE-Core-Library** 打开 KDE-Core-库文档的类索引文件。
- KDE-GUI-库 **KDE-GUI-Library** 对于 GUI-library ，如上。
- KDE-KFile-库 **KDE-KFile-Library** 对于 KFile-library，如上。
- KDE-HTML-库 **KDE-HTML-Library** 对于 HTML-library，如上。
- API **Project-API-Doc** 切换至项目的类-文档索引文件。

- 用户手册 **Project-User-Manual** 打开你当前项目的用户手册的索引文件。这可用于查看由 KSgml2Html 生成的 HTML-output 。
- 关于 KDevelop **About KDevelop...** 显示 KDevelop 的相关信息窗，包含以前的版本号，作者名和 email 地址及 Kdevelop 的执照参考。

## 5.5 工具条条目 Toolbar Items

KDevelop 提供通过工具条快速进入各种命令的途径。这些都是标准的及浏览器的工具条；在对话框编辑器 dialog editor 模式下只有标准工具条可见。 二者都可以通过"视图" 菜单中的相关菜单项来显示/隐藏；也可被拖出主窗口，在工作区的任一侧放置。

### 5.5.1 标准工具条 Toolbar

标准工具条提供进入最近使用的处理和编辑文件及建设你的应用的功能的快速途径。 按钮将从左到右执行以下命令：

- 打开项目 - 显示打开项目对话。
- 打开文件 - 显示打开文件对话，随后会弹出窗口，让你快速选择当前项目头及源文件。
- 保存文件 - 将当前打开的文件保存到磁盘。
- 打印文件 - 打开打印对话。
- (分割线)
- 取消 - 取消上次操作。
- 重做 - 重新执行上次取消的操作。
- 剪切 - 剪切当前所选内容。
- 拷贝 - 拷贝当前所选内容到系统剪切板。
- 粘贴 - 将当前系统剪切板中内容插入光标当前位置。
- (分割线)
- 文件编译 Compile File - 在源文件窗口可视化编译当前文件。在对话编辑模式由生成文件按钮代替。
- 生成 - 在项目上激活 make。
- 全部重建 - 重建项目

- 调试 - 为在工具窗口 Tools- window 用二进制应用调试 debugging 打开 KDbg 。
- 执行 - 运行应用的目标二进制
- 停止 - 取消当前程序
- (分割线)
- 对话框编辑器 - 切换至对话框编辑器模式。这里，按钮将由源代码编辑器 editor 按钮代替。
- 树视图 - en-/disables the treeview and works as a toggle button to display the current state
- 输出视图 - en-/disables the output view and works as a toggle button to display the current state
- (separator)
- 这是什么..? 帮助按钮 - 将光标变为问号，让你获取关于 KDevelop GUI 部件的信息。

### 5.1.2 浏览器工具条 Toolbar

浏览器 browser 工具条是另一类为浏览文件和文档 documentation 提供了很多有用命令的工具条。这一工具条包括:

- 类 combo box - 让你选择要浏览的当前项目中的某类
- 方法 combo box - 让你选择当前类的一个方法，浏览该方法的实现 implementation 。
- 类帮助按钮 - 单击鼠标,将你带至当前选中的方法的声明 declaration 处。包含一个弹出菜单以使用类工具增加类，方法，属性及浏览命令。
- 后退 - 在文档浏览器历史 documentation browser history 中向后浏览；包含一个弹出菜单以选择后退历史中的某一页。
- 前进 - 在文档浏览器历史 documentation browser history 中向前浏览；包含一个弹出菜单以选择前进历史中的某一页。
- 停止 - 停止浏览器装载一个文档文件的请求
- 刷新 - 重载当前显示的页面
- 返回 - 在浏览器中打开 KDevelop 用户手册索引页面
- 找寻标注文本 - 找寻选中文本的文档索引； 使用浏览器和编辑窗口 editor windows。

- 找寻关于...的帮助 - 打开找寻关于...的帮助对话，让你键入一个要在文档中找寻的关键字。

## 5.6 键盘快捷键

本节处理 KDevelop IDE 中使用的可设置的键盘命令预先定义的值及标准的取值。要得到一个关于如何改变命令的指定值的详细的解释，请参见 改变键盘快捷键 一节。

### 5.6.1 文本处理的快捷键

#### 5.6.1.1 光标移动

| | |
|---|---|
| 左移一个字母 | 左箭头 |
| 右移一个字母 | 右箭头 |
| 左移一个字 | CTRL+左箭头 |
| 右移一个字 | CTRL+右箭头 |
| 上移一行 | 向上箭头 |
| 下移一行 | 向下箭头 |
| 到行首 | POS 1 |
| 到行末 | END |
| 向上一页 | PageUp |
| 向下一页 | PageDown |
| 到当前文件开头 | CTRL+PageUp |
| 到当前文件尾部 | CTRL+PageDown |

#### 5.6.1.2 文本选择

| | |
|---|---|
| 左移一个字母 | SHIFT+左箭头 |
| 右移一个字母 | SHIFT+右箭头 |
| 左移一个字 | CTRL+SHIFT+左箭头 |

| | |
|---|---|
| 右移一个字 | CTRL+SHIFT+右箭头 |
| 上移一行 | CTRL+向上箭头 |
| 下移一行 | CTRL+向下箭头 |
| 到行首 | CTRL+POS 1 |
| 到行末 | CTRL+END |
| 向上一页 | SHIFT+PageUp |
| 向下一页 | SHIFT+PageDown |
| 到当前文件开头 | CTRL+SHIFT+PageUp |
| 到当前文件尾部 | CTRL+SHIFT+PageDown |

### 5.6.1.3 插入和拷贝文本, 制表

| | |
|---|---|
| 打开/关闭插入模式 | INS |
| 将选中文本拷贝到剪贴板 | CTRL+C, CTRL+INS |
| 从剪贴板插入文本 | CTRL+V, SHIFT+INS |
| 删除当前行 | CTRL+K |
| 在当前行后插入一行 | END, 再回车 |
| 在当前行后插入一行 | POS 1, 再回车 |
| 取消一编辑步骤 | CTRL+Z |
| 重做取消的步骤 | CTRL+Y |
| 制表 | TAB |

### 5.6.1.4 删除文本

| | |
|---|---|
| 删除光标左边的字母 | 退格键 |
| 删除光标右边的字母 | Delete |
| 删除选中文本 | 选中文本, 再回车或 Delete |

### 5.6.1.5 用编辑器 Editor 查找文本

| | |
|---|---|
| 打开到行...对话 | CTRL+G |
| 打开寻找文本对话 | CTRL+F |
| 重复上一次查找 | F3 |
| 打开查找/替代对话 | CTRL+R |
| 在文件对话中打开查找(文本查找工具) CTRL+ALT+F | |
| 用 Grep 查找标注文本 | SHIFT+F2 |
| 定位下一个错误 | F4 |
| 定位上一个错误 | SHIFT+F4 |

### 5.6.1.6 用文档浏览器搜寻文本

| | |
|---|---|
| 在文档中搜寻选中的编辑器文本 | F2 |
| 在文档中搜寻选中的浏览器文本 | F2 |
| 显示同一页中的下一个搜寻结果 | F3 |
| 在项目中搜寻选中的浏览器文本 SHIFT+F2 | |

### 5.6.1.7 浏览器快捷键

| | |
|---|---|
| 前一页 | ALT+ 左箭头 |
| 后一页 | ALT+ 右箭头 |

### 5.6.1.8 管理书签

| | |
|---|---|
| 加入书签 | CTRL+ALT+A |
| 清除书签列表 | CTRL+ALT+C |

### 5.6.1.9 工具条符号的快捷键 Toolbar Symbols

| | |
|---|---|
| "打开文件"符号 | CTRL+O |
| "保存文件符号" | CTRL+S |
| "打印文件"符号 | CTRL+P |
| "取消操作符号" | CTRL+Z |
| "重做"符号 | CTRL+Y |
| "剪切"符号 | CTRL+X |
| "拷贝"符号 | CTRL+C |
| "粘贴"符号 | CTRL+V |
| "编译文件"符号 | CTRL+F8 |
| "生成符号" | F8 |
| "执行"符号 | F9 |
| "用 Arguments 执行"符号 | ALT+F9 |
| "对话框编辑器"符号 | CTRL+D |
| "后退"符号 | ALT+左箭头，如果浏览器已打开 |
| "前进" 符号 | ALT+右箭头，如果浏览器已打开 |
| "搜寻标注文本"符号 | F2 |

### 5.6.1.10 窗口管理

要切换到某一窗口，请在窗口标题上按下 ALT 加带下画线的字母，例如 Tools 就是 Alt+T

### 5.6.1.11 编译过程的快捷键

| | |
|---|---|
| 编译当前源文件 | CTRL+F8 |
| 建立当前项目目标 | F8 |
| 在建立程序后执行目标 | CTRL+F9 |
| 用 Arguments 执行目标 | ALT+F9 |
| 中止当前程序 | F10 |

# 6. 帮助系统

大多数 Kdevelop 的精力都花在了帮助系统上。这应作为一个如何使用一套完整的帮助功能来扩展你自己的 KDE 应用的例子来理解，而且已经在 template 应用中部分实现，因此由应用向导生成的 KAppWizardKDE/Qt application frameworks 已经包含了为状态条帮助和文档提供的基本功能，那只需由程序员扩展即可。因此本节将介绍 IDE 提供的一般帮助的使用及帮助浏览器 Helpbrowser 的使用，巧妙的使用它将使你得到开发所需的信息变的十分容易。

## 6.1 "这是什么 ?"-按钮及快速帮助 Quickhelp

看看 Kdevelop 的顶部工具条，你会在最右边看见 "这是什么?" 按钮。选择这 一按钮，光标会变为一个指针，右边带一个问号，和工具条中的那个按钮一样。现在，你可以选择 Kdevelop 用户界面中的任何一个可视化部件。单击它会显示一个帮助窗口，为你简要描述该部件可为你提供的功能或它可以为你完成的任务，为主视，树视图，及工具条上的每一个按钮提供解说帮助。再次用鼠标单击或输入键盘上某键，这是什么帮助窗口将消失，你的鼠标会设置到上一次的位置。注意，如果你将鼠标指针放于某键上，该键会突起，接着出现一个快书-Tip-窗口，为你描述该按钮在菜单条中代表的功能；将鼠标指针移开窗口就会消失。

对于用户交互对话，快速帮助 Quickhelp 为你提供帮助窗口，简要描述你需要得到帮助的选中条目。这可通过在项目上右击鼠标来进入，一个内容目录会弹出,允许选择"快速帮助"。选中它会弹出帮助窗口。这在你还对 Kdevelop 不熟悉，不知道某操作对话的目的时很有用。注意，大多数对话提供一个帮助按钮，在帮助浏览器的在线手册中为你提供对话选项的详细文本帮助。

## 6.2 状态条 Statusbar 帮助

Kdevelop 的状态条帮助提供给你很多的功能，告知你当前操作状态的相关信息并显示为你提供命令简要描述的"状态信息"。

### 6.2.1 状态条项目 Statusbar Entries

状态条 Statusbar 包括:

1. 一个一半信息域，多数居左。常用于帮助信息及显示当前操作。
2. 进展条，指示需要相当长一段时间才能完成的操作的进展，例如保存文件及类浏览器的扫描过程。进展条只有在那样的程序执行时才会出现。
3. 一个插入/改写指示器。它将显示编辑器 editor 模式，对于插入操作将由 INS 表示插入模式，改写模式则使用 OVR 。也可通过键盘上的 INS-键转变模式。
4. 一个行计数器，显示光标当前所在的行。
5. 一个列计数器，指示光标当前在一行中所在的列位置。

### 6.2.2 帮助信息

当你选择了菜单条中某菜单的一个操作但并未执行，状态条告诉你关于该菜单条目的操作。此外，如果你按下了工具条按钮，但还未松开鼠标键，该按钮的帮助信息会以和菜单条目相同的方式显示。你可以一直按住鼠标，通过将鼠标指针移开该按钮来阻止所选按钮的执行。一旦光标移开，鼠标按钮即可松开。

对于当前正执行的操作，Kdevelop 会显示其执行过程。这对在后台运行的程序其作用，例如保存文件及对话。如果程序已经存在，例如一个 make-invocation，状态条会将改变显示回"Ready"状态。而且，在使用文档浏览器 documentation browser 时，只要将光标置于浏览器窗口中一个 URL 连接上，状态条会显示连接地址。因此，你可以轻易的发现文件是本地的还是只有通过网络连接才能访问的远程文件。

Kdevelop 项目编辑器的字表明了当前在顶部窗口打开的文件的名称。这可能是文档浏览器 documentation browser 的一个 HTML 文件，或编辑窗口的一个文本文件。此外还将显示项目名，这样你可以一直控制你的位置及你当前的操作对象。

## 6.3 配置 HTML 浏览器

Kdevelop 中包含的浏览器是完全基于 HTML 的，因此你可以指定一般选项，例如背景等。你可以通过"选项菜单的"文档浏览器"项设置所有的参数。配置对话 configuration dialog 未你显示两个定位装置；第一个是进行字体参数设置，第二个是颜色选项设置。

### 6.3.1 字体参数

字体显示的第一个选项是字体大小。可选项由小号，中号和大号。一般用途的最佳显示缺省设置为小号。

对于字体选择，你必须为普通文本指定显示在 HTML 文档中的标准字体；固定的字体是用于例如在 HTML 文件中显示代码的。

### 6.3.2 颜色参数

颜色参数对话允许设置背景，标准文本, URL 连接及连接后的颜色。右端的有色按钮显示的是当前设置；选择一个按钮将打开"颜色选择"对话。在那里，你可以通过在多色窗口中选择一个系统色，一个自定义色,或通过直接设置颜色值来指定颜色。中部是对所选颜色的预览。另外，你可以指定浏览器是否应该将连接下划线以更醒目及你是否希望使用你自己选择的颜色，而不管页面预先设置的颜色。"应用" 会执行所有的改动, "确定"则运用所有的改动并关闭对话。"取消" 不会改变任何设置并退出配置。

## 6. 4 使用文件浏览器

文件浏览器 documentation browser 允许快捷的访问随 Kdevelop 提供的或自动生成的所有的手册及文件，包括 KDE 库 KDE-libraries 及你的项目文件的在线文档。另外，树视图中的文件树允许你设置一个额外的包含了所有个人加入文件的"其他" 文件夹。

### 6.4.1 要求

要使用帮助浏览器 Helpbrowser 的所有特色，你应该线安装 KDoc 及 glimpse。KDoc 将在 setup 过程中为 KDE 库 KDE-libraries 生成各种各样的在线类--文件，但也可通过 KDevelop Setup 中的选项激活。对于一个你的项目的类 classes 的 API (Application Programming Interface)文件的生成，KDoc 也会被项目菜单中"生成 API-Doc"使用和访问。这将处理项目所有的当前的头文件及对 Qt 和 KDE-libraries 前后参照，如果它们在帮助浏览器 Helpbrowser 中可用。对于项目手册的生成，你应该至少在你的系统上安装 KSgml2Html (由 KDE-SDK 提供)及, SGML-tools。要扩展及更改你的项目的在线文件，你必须通过在 RFV 中选择它来为你的项目编辑

文档文件。在保存了你的改动后，从项目菜单中访问"建立用户手册" 。假如 SGMLtools 检测到格式化错误 errors，错误将会显示在输出 output 窗口，允许你直接找出错误行。"glimpse"程序用于自动为你的文件创建一个个人的搜寻索引。索引通常在安装过程中建立，但也可利用 KDevelop Setup 对话建立。要得到所提供的搜寻功能的描述，请参见 使用搜索索引。

## 6.4.2 提供的文档

KDevelop IDE 有两套在线文档，可以通过帮助菜单或在 Kdevelop 文件夹中通过树视图的文档树来获得。所提供的第一本书就是这本包含了所有你需要的安装及设置，可用功能及应用开发介绍信息的在线手册。第二本书的首版是 KDevelop 编程手册。编程手册包括与创建及扩展由 KDevelop 生成的项目相关的大多数问题。通过此教程，用户得到关于由 Qt- 和/或 KDE-libraries 创建的应用所提供的丰富功能的介绍，还为保证 KDE-compliance 提供了指导,在位于 http://developer.kde.org 的 Internet 网站上提供相同介绍。然而，编程手册不能代替任何其他可用的关于 C++编程语言打印出的或是电子形式的文件，及关于 Qt-库的使用的文件。

KDevelop 使用的 C/C++- 参考 Reference 当前只有在位于 http://www.kdevelop.org 的 Kdevelop 主页上可用。在发布时该参考可能会包括进去，可参见所发布的安装程序索引以获得更多信息。它可通过下载和拷贝源文件到主要的 KDE-目录($KDEDIR)来轻易安装。然后，你必须使用 tar zxvf c_c++_reference.tar.gz 将它作为根部，然后参考将被拷贝 KDevelop 的文件目录。要卸载参考文件，你只需在($KDEDIR)/share/doc/HTML/default/kdevelop/reference 下删除"参考"文件夹。

Qt/KDE-libraries 文件夹允许直接访问你的 Qt-库拷贝的 HTML-在线文件。库文件的路径通常使由 Kdevelop 安装程序自动检测，但也可在 KDevelop Setup 对话中手动设置。而且，所有 KDE 库 KDE- libraries 的可用文件都以库名顺序列出，因此如果你要使用某个库的类 classes，你可以很容易的决定要加入项目中连接器 Linker 设置的库的类型。注意，整个 KDE- 库文件只有当由 KDoc 生成时才可进入---因此包含于 KDE-SDK 中的这个程序，必须文件生成以前安装。由于 Kdevelop 的安装程序自动完成次步骤，它应当在运行 KDevelop 安装程序前安装。如果不是这样且文件不能建立，你可以利用 KDevelop Setup 对话在其后任何时间创建它。文件树还包含"其他" 文件夹，这些文件夹旨在包含所有个人设置的文档，如下。最后，文档树允许访问 API 及你当前项目的手册。

### 6.4.3 将文件加入帮助浏览器 Helpbrowser

要设置你的帮助浏览器 Helpbrowser, 先打开树视图并选择"DOC"定位装置。你会看见一个打开包含四哥文件夹的树。"其他" 文件夹缺省为空。用鼠标在这个文件夹上右击, 一个上下文菜单会打开，它包含了一个"增加入口"项。选择它会打开增加入口对话，你必须在此输入两个值: 上面一个是文档在文档树中显示的文件名，下面一个是文件入口行, 你必须在此输入其后通过选择该入口将被打开的起始页的路径和文件名。你可以直接输入路径及文件名或通过选择右边的按钮来打开一个文件选择对话。这允许对你的系统进行快速浏览以查找你的起始页的路径。注意，只有 HTML-文件可以选择, 所以只允许选择 HTML 文件。选择 OK 会将入口加入文档树并可直接使用。

### 6.4.4 使用搜寻索引

KDevelop 包含了一套功能，帮助你利用 HTML 文件查找信息。要使用这些特色, 你的系统需要安装"glimpse"程序, 一个免费的数据库生成器-它会生成搜寻索引 search index 并在文件内执行搜寻操作。要建立索引, 请参见 KDevelop Setup 。索引也可在安装程序中自动生成。

如下，有几种方式可以进入查寻功能，你可以选择:

1. 在编辑器 editor 中, 选择你想要获得帮助的文本活将光标置于你要查寻的单词中。然后右击鼠标选择设置: "语句"。这也可通过从帮助菜单中选择"搜寻标注文本"、按下快捷键 F2 或通过从工具条选择搜寻按钮来完成。
2. 在文件浏览器中选择你要找寻的文本并按下鼠标右键, 选择找寻:" 语句" 或选择 "找寻标注文本" 或如上使用找寻按钮。
3. 如果你想要查找一个特定的关键字, 在帮助菜单中选择"查找对...的帮助" 或从工具条中选择"查找对...的帮助" 按钮, 然后会打开一个查找对话，你可以在此输入你想要得到信息的词句。

可通过在"DOC" -树-浏览中选择相关按钮或通过帮助菜单的条目直接进入特定文件。

帮助浏览器 Helpbrowser 还提供对文本查找工具 grep 在你当前的文档中查找所选文本的支持, 例如，你已经为 KTMainWindow class (KDEUI library)打开文档

页并希望知道 KTMainWindow 出现在你代码中的哪个位置。标注 KTMainWindow 并选择 "文本查找工具: KT 主视窗"或按下 Shift+F2。"在文件中查找"对话会打开并直接显示你的文本查找工具 grep 的搜寻结果。然后你可以选择一个结果行并跳至相关资源代码。

# 7. 使用编辑器

本集成开发环境的一个重要部件就是编辑器 editor。你可以利用它完成以下操作:

- 生成, 打开并保存资源及项目文件 project files
- 编辑资源及项目文件 project files
- 编写你的 SGML 文件
- 打印 printing 你的项目文件 project files

通常,与其他编辑器相比，该编辑器 editor 本身并没有多大的不同；尤其它是流行的 Kwrite 的内置版本, 也被称作"扩展的编辑器"。 如果你对这个很熟悉, 你对使用它来管理你的项目不会有很大问题。除了常规编辑器, Kdevelop 还包含一个新的打印 printing 系统, 它有进一步的扩展以利于更好使用的资源代码。--你可以选择你想要使用的打印 printing 程序。

本节将进一步为你介绍关于如何处理你的项目文件 project files 及让你更加熟悉编辑器 editor 的功能。许多 Unix-支持者更愿意使用 Emacs 或 XEmacs- 他们对这个可以运用自如。但是对于简易编程则没有必要使用它, 尤其是对初学者, 使用功能强大的编辑器，最后往往只允许你输入你的代码。

## 7.1 管理项目文件

以下几节将描述如何生成, 保存, 打开并关闭你需要编辑的项目文件 project files- -注意这并不包括翻译文件或 pixmaps。那些是自动识别且相关编辑程序，例如 KTranslator 会为你打开那些文件。

为了让你能更轻易的打开你的文件, Kdevelop 包含了两个易于操作的树视图, 类似于一个文件管理器, 如果你选择一个文件就可识别, 并在相关的编辑器 editor 窗口打开它。主视包含两个独立使用的窗口, 但它们通过项目编辑器相连, 因此你同样可以完成所有菜单条操作。这样做的目的是允许你在同一时刻对两个窗口进行操作, 虽然只有一个可视。 C/C++ 窗口则有另一个任务, 允许你一个一个的轻易完成对资源的编译, 这样你就可以检测它的执行而无须在你的项目上运行一个完整的建立程序了。

当 C/C++ 窗口在顶部时就可以这样做。请从建立菜单中选择"编译文件 Compile File" 或按下工具条中的相关按钮，你的文件就会保存并编译, 你也可以

在输出 output 窗口中通过编译器 Compiler 输出 output 来控制任何错误 errors 。

在 IDE 内你可以随你所意打开任意多个文本文件。所有打开的文件都在窗口菜单中列出，这样你就可以通过在菜单中选择相关文件名来进行切换。此外，实际打开的文件将在 Kdevelop 的窗口框架上显示其文件名。

## 7.1.1 生成和保存文件

要生成一个新文件，请从文件菜单中选择"新建"。 这会打开"新建文件 New File"对话，你可在此指定文件名和类型。另外，你还必须设定目的目录及文件是否被加入项目。最后，万一你想要加入新的资源文件而不由类生成器生成一个新类，你还可以使用你的项目的头部模板。在文件创建以后，你可以象平常一样编辑此新文件；如果你必须在不同的文件间切换，你总是可以通过文件树或窗口菜单切换回原来的文件。

对于保存变动，Kdevelop 提供了各种各样的选项。标准的方式是通过从文件菜单中选择"保存"或 "保存为"或按下工具条的保存按钮来保存文件。要想一次保存所有改动过的文件，你还可从文件菜单中选择"全部保存"。

现在，当着手一个项目时，如果在你对文件有很多的改动时发生了什么紧要的事，是十分恼火的；有时你甚至会因为自己忘记保存文件的改动而狠狠责怪自己。Kdevelop 会通过提供的"自动保存"来关照此事，这是缺省进行的，每 5 分钟会保存一次所有的文件。要选择另外的保存时间间隔或激活此功能，请参见 KDevelop Setup 以查看对设置 setup 选项的介绍。

另外，KDevelop takes care for all changes if you open another project or exit KDevelop. 你将被询问是否保存改动过的文件，你还在此可以选择并指定你想要保存或不保存的文件。此外，当激活任何建立过程时，你的文件会自动保存，因此你不会奇怪为什么在你改变为资源文件以后，你的应用并未象你所希望的那样去运行。唯一的例外是"编译文件 Compile File"命令，它只保存当前在资源编辑器 editor 窗口中以可视化方式打开的资源文件。

## 7.1.2 打开和关闭文件

要打开一个资源文件，你也有很多选择。其中一种是编辑人员的标准方式，即从文件菜单中选择"打开"。你会看见一个"打开文件"对话，允许你选择你想要编辑的文件。另外一种也许更为常用的方式是从 LFV 中选择文件，逻辑文件浏览器，或 RFV，实时文件浏览器(参见 概要)。文件树的优点是它们提供了到你的 ""

一个快速可视的?路，""LFV,它只显示通过类型你的项目文件 project files，并搜集于文件夹中。你也可以通过在文件树上单击鼠标右键以另外一种方式为你的文件分类来设置 LFV。从弹出菜单中选择"新组"，然后你可以设置一个新的类，或通过指定组的由逗号隔开的文件扩展名来选择"编辑类"以安装新文件过滤器。一个由 KAppWizard 生成的项目的标准文件组是头部，资源，GNU 及其他。另外，在增加了一个翻译文件以后，Kdevelop 增加了一个"翻译"文件夹，它包含了你的*.po 文件。

要关闭文件，请从窗口菜单中选择要关闭的文件，这会将打开的文件装载到前部的编辑器 editor。然后从文件菜单中选择"关闭"。如果你的文件有所改动，你会被询问是否保存它。当关闭项目时，所有当前打开的文件会被检测是否有所改动，然后询问你是否需要保存。

## 7.2 在文件中导航

以下将大体性指导你如何在你的文件中进行定位以更快找到你需要的东西。

### » 如何找到文件中的某行

1. 在视图菜单中选择"到。。。行" 或按下 CTRL+G。对话区域"到。。。行"会出现。
2. 插入你要去的行号。
3. 按下确定。

### » 如何设定书签

1. 将光标设置于你想要通过书签进入的行。
2. 从"书签"菜单中选择"设置书签"。
3. 一个弹出菜单会允许你想要为新书签设置的书签号。
4. 选择书签号码。

另一种设置书签 bookmarks 的方式是从"书签"-菜单中选择"加入书签"。这会将书签设置给当前行并填入书签 bookmarks-列表。文件浏览器 documentation browser 还提供通过上下文菜单条"加入书签"给当前页设置一个书签。

### » 如何删除书签 bookmarks

书签 bookmarks 是为每个编辑窗口单独设置的---注意你的书签 bookmarks 逼供不和你为它设置的某文件相连。要删除所有书签，请从书签菜单中选择"删除

书签"。这会删除实际位于顶部的窗口的书签 bookmarks ，不管是头部窗口，C++窗口还是浏览器窗口。

» 如何到一个设置了书签的行

1. 选择"书签"菜单，打开包含了你想要浏览的书签窗口的书签 bookmarks 的弹出菜单；不管是 C++窗口还是头部窗口入口。
2. 选择设置了书签的行。

为浏览器窗口选择书签将打开浏览器并装载页面。

## 7.3 利用键盘快捷键操作

使用编辑器 editor 时，你应该尽量熟悉一些键盘快捷方式 shortcuts，这会使得放置光标编辑文件更加容易。完整的快捷方式参考列表于 概要。

---

| | |
|---|---|
| 左移一个字母 | Left Arrow |
| 右移一个字母 | Right Arrow |
| 左移一个字 | CTRL+Left Arrow |
| 右移一个字 | CTRL+Right Arrow |
| 上移一行 | Up Arrow |
| 上移一行 | Down Arrow |
| 到行首 | POS 1 |
| 到行末 | END |
| 向上一页 | PageUp |
| 向下一页 | PageDown |
| 到当前文件开头 | CTRL+PageUp |
| 到当前文件尾部 | CTRL+PageDown |
| 左移一个字母 | SHIFT+Left Arrow |
| 右移一个字母 | SHIFT+Right Arrow |
| 左移一个字 | CTRL+SHIFT+Left Arrow |
| 右移一个字 | CTRL+SHIFT+Right Arrow |
| 上移一行 | CTRL+向上箭头 |

| | |
|---|---|
| 下移一行 | CTRL+向下箭头 |
| 到行首 | CTRL+POS 1 |
| 到行末 | CTRL+END |
| 向上一页 | SHIFT+PageUp |
| 向下一页 | SHIFT+PageDown |
| 到当前文件开头 | CTRL+SHIFT+PageUp |
| 到当前文件尾部 | CTRL+SHIFT+PageDown |
| 打开/关闭插入模式 | INS |
| 将选中文本拷贝到剪贴板 | CTRL+C, CTRL+INS |
| 从剪贴板插入文本 | CTRL+V, SHIFT+INS |
| 删除当前行 | CTRL+K |
| 在当前行后插入一行 | END, 再回车 |
| 在当前行后插入一行 | POS 1, 再回车 |
| 取消一编辑步骤 | CTRL+Z |
| 重做取消的步骤 | CTRL+Y |
| 制表 | TAB |
| 删除光标左边的字母 | 退格键 |
| 删除光标右边的字母 | Delete |
| 删除选中文本 | 选中文本, 再退格 |

## 7.4 编辑窗口设置

Kdevelop 内的编辑器 editor 可以为特殊编辑需要设置为对所有编辑窗口都有效。这样你设置颜色模式，高亮颜色(还可参考文件的编程语言来设置)及自动文本设置，例如 tab- 并选取模式。以下将叙述如何通过"选项"菜单提供的设置对话设置这些选项。

### 7.4.1 一般设置

编辑器 editor 的一般设置可以用选项菜单中的"编辑"来设置。选择相关参数值并在你完成设置后按下确定。

### 7.4.2 编辑选项

**自动缩进:**

这将编辑器设置为当键入一个新行时将光标放于第一行文字之下。

**回删时对齐:**

当退格键按下时,这一选项将光标设置于上方第一行文字之下.

**自动换行:**

在"每行字符数:"中设置了后，文字将置于下一行

**替换制表符:**

当前文本中的定位装置由"制表符号宽度:" 值代替

**删除后缀空格:**

去除其后的空格

**光标对齐:**

当退格在行首输入时，将光标置于最后一行的尾部。

**自动加括号:**

当括号(任何一种)打开时，在光标前生成一个反括号。

### 7.4.3 选择选项

**持续选择:**

当把光标置于其他位置时，选取的选项保持选中

**多行选择:**

允许在文本种进行多种独立的选择。

**垂直选择:**

允许文本的垂直选择。

**输入时删除:**

在编写选择时删除一个选择。

**Toggle Old:**

只允许一种选择。当右另一种选择时会取消先前的选择。

**每行字符数:**

设置一行可包含的最大列数。超过该值的包含了一个字母的单词会自动断到下一行。

## 7.4.4 颜色

要改变编辑器 editor 通常的面貌, 你可以通过从"选项"菜单中选择"编辑器颜色"定义编辑器使用的一套颜色。你为以下项目可以设置颜色:

- 背景 **background:** 编辑器的背景
- 文本背景 **text background:** 显示的文本的背景
- 选中 **selected:** 选中文本的颜色
- 找到 **found:** 通过"编辑"—"查找", "重复查找"及"置换" 菜单在查找中找到的文本的颜色。
- 选中并找到 **selected + found:** 选中要查找及找到的文本的颜色

## 7.4.5 语法高亮

Kdevelop 编辑器 editor 的句法高亮模式可以通过两个对话设置; 第一个你可以通过" 选项"-菜单中的"编辑器缺省"入口为句法高亮设缺省颜色。在那里, 你可以社会字颜色, 字体及字体大小, 例如关键字。选择缺省项目并设置所有需要的选项。

第二个设置对话 configuration dialog 可通过"选项"-菜单中的"句法高亮"入口进入。这允许你为编程语言设置文件过滤器, 例如 C++的*.cpp, *.h。然后选择你

要设置的项目。如果你希望使用你已经在"编辑器对话" 设置的缺省值，请选择项目风格及项目字体选区的"缺省"框。这会为选中项读除缺省设置。在按下确定后，你新的值即生效并由编辑器 editor 使用。

# 7.5 查找和替换

## 7.5.1 单个文件查找

**» 如何在实际编辑窗口找到一个或更多字符**

1. 从"编辑"-菜单中选择"查找"。这会打开查找对话。
2. 在编辑区域指明要查找的语句。下拉菜单会提供选择前一次查找的语句。
3. 选择另外的选项，例如"只查找整个单词"
4. 按下"确定"。

要重复查找键入查找对话中的一个语句，按下 F3。

## 7.5.2 在多个文件中查找

由于查找函数只涉及当前可见的单个文件中的查找，你的局限性很大。但你常常想在你的整个项目中查找同一语句。因此，Kdevelop 包含了一个文本查找工具 grep-对话，让你在你通过设置开始查找的目录和/或 mime-类型来指定的所有文件中进行查找。因此，指定确切目录及 mime-类型将减少 Kdevelop 阅读你的文件及显示结果所需要的时间。要开始在多个文件中的查找，请从"编辑"-菜单中选择"在文件...中查找"。查找对话会打开并让你输入:

- 要查找的文本(样式)
- 用于查找的模板
- 要查找的文件的模拟类型
- 开始查找的目录
- 如果查找在所有包含的子目录中是递归的

在缺省状态，文本查找工具 grep-对话设置为在你的项目目录下启动并在实现 implementation 和头文件下递归的工作。

你甚至可以通过使用以下选项扩展你的查找样式:

1. **.** 匹配任何字符
2. **^** 匹配行首
3. **$** 匹配行尾
4. **\<** 匹配词首
5. **\>** 匹配词尾

要想重复搜寻，你还可以使用可用的运算符:

1. **?** 前一个项目匹配一次以下
2. ***** 前一个项目匹配为 0 或多次
3. **+** 前一个项目匹配一次或多次
4. **{ n }** 前一个项目匹配正好 *n* 次
5. **{ n,}** 前一个项目匹配 *n* 次或更多次
6. **{,n }** 前一个项目匹配少于 *n* 次
7. **{n,m}** 前一个项目匹配至少 *n* 次但少于 *m* 次

括号内的语句的参考通过注释*****n* 可用。

在指定好你的查找后，按下"查找"。结果将显示在结果窗口。要跳至某文件或行号，请选择结果行并按下回车或双击该结果。编辑器将自动打开相关文件并将光标置于结果行。这允许完全指定任何查找操作并给除确切结果。

Kdevelop 还提供一些更专业化的函数来使用 editors 和浏览器中的文本查找工具 grep。在任何一个窗口中选择你要查找的语句并按下 SHIFT+F2，或从右击鼠标弹出的菜单中选择 "grep:<your_expression>"。这将请求文本查找工具 grep 在你的项目目录的文件中查找这一语句并立即显示结果。如上所述切换至结果。从编辑器窗口内，将光标放置于一个单词上并开始查找；光标下的词即查找样式。

## 7.5.3 文件中查找

着手一个项目时，你常常需要关于你要使用的元素函数 member functions 的参数- 常常是你符合你需要的函数名，但是参数是很难记的。因此，也由于其他一些可能的目的，Kdevelop 包含了一个查找函数，结合文件浏览器 documentation browser 来查找出现在你的文件中的语句。要使用这以查找函数，你需要先正确设置文件浏览器 documentation browser 并创建查找数据库 。要激

活文件中查找，请按以下步骤:

1. 将光标置于你要搜寻或标注语句的文字中
2. 从帮助菜单中 Help-menu 选择"查找标注文本" 或按下鼠标右键以打开上下文菜单；再选择"查找: "语句"。
3. 在查找结果 search result 页面显示在文件浏览器 documentation browser 中后，选择你认为可能包含你需要的信息的页面。
4. 选中文件页面将显示，你的查找结果 search result 也会标注。要显示同一个文件页面中的下一个查找结果，请按下 F3。

这允许你轻易找到你想要的信息。要使用结果，文件浏览器 documentation browser 允许选择并拷贝到剪贴板。然后你正在编辑的文件并从"编辑"-菜单选择"粘贴"。

要得到关于如何使用文件的详细描述，请参见 使用搜索索引。

### 7.5.4 替换文本

要替换一个语句，请从"编辑"-菜单中选择"查找并替换"。 "查找并替换"-对话允许你指定要替换的语句及要替换为的语句。接着按下"确定"。第一个找到的语句将被标注，这样你就可以看见该语句的位置及所在文本。然后你可以通过一个对话指定是否该语句被替换。当查找到达当前文件尾部而结束后，你会被询问是否愿意从头开始查找。如果你已经完成，请选择"取消"。

## 7.6 打印

由于 Kdevelop 是设计来为开发者提供进入文件及信息的最好的通路和减少开发周期，它还包括了一个新的打印 printing 设备，它使用了两个在 Unix-系统可用的一般的打印 printing 程序，即 *a2ps* (ASCII-to-Postscript)和 *enscript*. 除此之外，你还可以通过直接使用 lpr (行式打印机设备) 来打印。由于使用 a2ps 或 enscript 提供了大多数打印 printing 的选项，你应该在你的系统上先安装任意一个；两个程序通常都随发布的软件发布，因此你可以毫无问题的得到它们。但是在打印 printing 之前，你应该先看看可用的设置对话 configuration dialogs 来根据你的需要准备号打印的输出 output。下面的几节将描述如何设置 Kdevelop 来打印 printing 文件。

## 7.6.1 设置打印机

打印 printing 程序可以通过从文件菜单"File"-menu 中选择"打印"；在打印 printing 对话中，通过左上角的下拉菜单选择程序。然后按下右边的"选项"按钮。这将打开选中要使用的程序的设置。设置程序的另一种方式是从"选项"菜单中选择"打印机设置..."；然后选择"a2ps" 或"enscript"。

## 7.6.2 设置选项 a2ps Configuration Options

### 7.6.2.1 打印

- 头部 **header:** 将一个头部框架加入页面
- 文件名 **filename:** 如果选中，头部框架以文件名作为器内容
- 登录 **login:** 将用户 ID 加入页面右上角
- 边界 **borders:** 为下一页加入框架边界
- 日期及时间 **Date & Time:** 加入打印 printing 日期及时间
- 排列文件 **align files:** 在同一页上打印几个文件，可用于双页打印 printing 模式
- 设置标签尺寸 **set TAB size:** 为打印 printing 标签设置标签尺寸
- 头部文本 **headertext:** 当文件名取消选中并允许插入另一个文本到头部框架时可用
- 字体大小 **fontsize:** 为文本设置字体大小。缺省的字体大小是 9

### 7.6.2.2 文本打印 printing

- 分裂行 **cut lines:** 如果以行的内容太长无法打印，则将行的内容切断。如果取消选中，行将被分裂。
- 解释 TAB,BS 及 FF**interpret TAB, BS and FF:** 解释标签，退格及快速前进字符。
- 替换不可打印的字符 **replace non-printing character by space:** 如果文件包含不可打印的字符，这些字符将以空格字符代替。
- 以 ISO-Latin 1 打印非 ASCII 字符 **print non-ASCII character as ISO-Latin 1:** 以 ISO-Latin 1 输出 output 模式打印出未包含于 ASCII 格式的字符
- 粗体 **bold font:** 以粗体打印出整个文本。

### 7.6.2.3 记数

- 行记数 **numbering lines:** 激活后从上到下为所有行记数
- 页记数 **numbering pages:** 允许通过以下方式选择页记数模式:
    - 单个文件 **file single:** 从 1 开始为每个文件页面记数
    - **file together:** 添加到第一个记数页所有后面的页面
- **lines per page:** 设置打印时每页最大的行数

## 7.6.3enscript 设置选项

### 7.6.3.1 头部

- 奇特的头部 **Fancy Header:** 增加一个奇特的头部
- 头部文本 **Header Text:** 允许加入头部文本
    - 文本 **text:** 设置文本内容
    - 位置 **position:** 设置文本内容位置为左/中/右
- 登录 **Login:** 将用户 ID 加入头部
    - 登录 **login:** 允许加入用户 ID
    - 位置 **position:** 设置用户 ID 的位置
- 文件名 **Filename:** 将文件名加入头部
    - 文件名长度 **Size of filename:** 以全名或简写加入文件名，意即完整路径或只是文件名
    - 位置 **Position:** 设置文件名的位置
- 主机名 **Hostname:** 为头部加入主机名
    - 主机名 **hostname:** 允许加入主机名
    - 主机名长度 **size of hostname:** 设置主机名的长度
    - 位置 **Position:** 设置主机名的位置

### 7.6.3.2 日期与时间

- 当前日期 **Current Date:** 包含当前日期
    - 当前日期 **current date:** 允许加入当前日期
    - 位置 **position:** 为日期条目设置位置
    - 格式 **format:** 设置日期格式

- 修改日期 **Modification Date:** 包括上一次修改的日期
    - 修改日期 **modification date:** 允许加入当前日期
    - 位置 **position:** 为日期条目设置位置
    - 格式 **format:** 设置日期格式
- 当前时间 **Current Time:** 包括当前时间
    - 当前时间 **current time:** 允许加入当前时间
    - **AM**PM:/ 使用 AM/PM 或 24h 格式
    - 位置 **Position:** 为时间条目设置位置
    - 格式 **Format:** 设置时间格式
- 修改时间 **Modification Time:** 包括上一次修改的时间
    - 修改时间 **modification time:** 允许加入时间修改
    - **AM**PM:/ 使用 AM/PM 或 24h 格式
    - 位置 **Position:** 为时间条目设置位置
    - 格式 **Format:** 设置时间格式

### 7.6.3.3 页面布局

- 记数及边界 **Numbering & Border:**
    - 行记数 **numbering lines:** 为打印 printing 加入文档的行号
    - 边界 **borders:** 为打印 printing 加入页面边界
    - 页记数 **numbering pages:** 为打印 printing 加入页记数
    - 文件排列 **align files:** 为页记数添加文件
    - 每页行数 **lines per page:** 每页最多行数
- 格式与标签 **Format and TAB:**
    - 设置标签尺寸 **set TAB size:** 为解释执行 TABs 设置 TAB 尺寸
    - 头部字体 **font for header:** 为头部文本设置使用的字体
    - 正文字体 **font for body:** 设置正文字体(文件内容)
- 文本打印 **Textprinting:**
    - 断行 **cut lines:** 断开太长的行。未选中则行将断开
    - 由空格取代非打印字符 **replace non-printing character by space:** 由空格取代打印 printing 字符集不支持的字符
- 其他选项 **Other Options:**
    - 目录表格 **table of contents:** 增加包含关于打印文件, 页记数等信息的目录页的表格。

- 高亮条 **Highlight bars:**
  - 高亮条 **highlight bars:** 打印 printing 时高亮条的行
  - 改变周期 **cycle fo change:** 设置改变高亮模式的行数
- 包裹的行 **Wrapped line:**
  - 标注包裹的行 **mark wrapped lines:** 分裂的行在打印 printing 时标注
  - 包裹的行的值 **value for wrapped line:** 原来的行分裂为新行的预设值

#### 7.6.3.4 下划线

- 文本 **Text:** 设置要下划线的文本
- 位置 **Position:** 设置下划线文本的位置
- 字体 **Font:** 设置用于下划线的字体
- 角度 **Angle:** 设置下划线文本的角度
- 灰度 **Gray scaling:** 设置下划线文本的灰度
- 风格 **Style:** 设置下划线文本打印为加边框或填充

### 7.6.4 打印对话

#### 7.6.4.1 直接打印选项

- 程序 **Program:** 设置打印 printing 要使用的打印 printing 程序: a2ps, enscript 或 lpr
- 打印机 **Printer:** 设置打印 printing 要使用的打印机
- 输出定位 **Output location:** 为打印 printing 到一个文件, 选择输出定位 output location
- 方向 **Orientation:** 设置打印 printing 方向未横向或纵向
- 拷贝 **Copy:** 设置每页要拷贝的数量
- 纸张大小 **Paper Size:** 设置要使用的纸张大小
- 输出格式 **Output Format:** 如果使用 enscript 作为打印 printing 程序, 你可以选择附言或 html 打印 printing
- 缺省打印设置 **Default Printsettings:** 设置打印 printing 时使用的缺省值
- 打印 **Outprinting:**

- 页打印 **Page Printing:** 设置在一张纸上打印一页或两页
- 页 **Pages:** 选择使用 enscript 的页为所有页，奇数或偶数页
- 精细打印 **Pretty Print:**
  - 精细打印 **pretty-print:** 允许 enscript 的精细打印模式
  - 颜色 **color:** 打印 printing 使用的颜色
  - 精细打印模式 **Pretty Print Mode:** 根据你的文件格式设置打印 printing 模式

### 7.6.4.2 文件选择

文件选择对话可以通过打印 printing 对话上的"文件"按钮来进入。文件的选择允许以一定的标准指定要打印出的文件:

- 文件选择 **File Selection:**
  - 当前 **current:** 当前打开的在编辑窗口可见的文件
  - 项目中所有文件 **all in project:** 当前打开的包含于项目的所有文件
  - 自选文件 **self chosen files:** 允许通过文件选择选取要打印的文件
  - 所有 cpp 文件 **all cpp files:** 打印出所有项目的资源文件
  - 所有头部 **all headers:** 打印出项目所有的头部文件
  - 改动的文件 **changed files:** 允许指定在一个时间跨度内改动的文件:
- 改动的文件 **Changed Files:**
  - 之间 **Between:** 指定在。。之后所有改动的文件:
    - 日期 **Date:** 改动文件时的日期
    - 时间 **Time:** 改动文件时的时间
  - **And:** 指定所有在。。。之前改动的文件:
    - 日期 **Date:** 文件改动的日期
    - 时间 **Time:** 文件改动的时间
- 自选文件 **Self Chosen Files:** 当自选文件被选中时可用(见上)
  - 增加 **add:** 按下它以增加一个要打印 printing 的文件，在行编辑区域
  - 删除 **delete:** 从打印 printing 列表中删除一个选中的文件
  - 清除 **clear:** 清除打印 printing 列表

### 7.6.4.3 打印预览

打印预览让你控制输出 output 的样子。因此, Kdevelop 使用 *ghostview* 或 *kghostview* 程序。当按下任一个打印 printing 对话中的预览按钮时, 你将看到一个输出 output 模板，为你显示当前设置的选项的效果。

# 8. 项目

## 8.1 项目类型

### 8.1.1 程序

KDevelop 以.kdevprj 结尾创建了一个项目文件。该文件包含你的项目的所有信息，因此小心不要将它删除了。它存储在项目的基本目录下，在装载项目时必须打开。项目文件保存了你的例如文件属性，安装路径，发布状态及编译选项 compiler options (CXXFLAGS) 的的文件的所有信息。文件属性设置允许你记录文件位置线索。

有了应用指南 KAppWizard，你可以根据你选择的应用类型创建一个心的应用项目。眼下，应用指南 KAppWizard 可生成四种应用框架，如下:

- 单一的文件界面(SDI) KDE-应用包含一个菜单条，一个工具条和一个状态条。它包含基本控制资源管理以允许将应用框架扩展为一个独特的 KDE 应用。应用框架还包含状态条帮助信息。从编程人员的眼光来看,它基于三个依靠 MVC-concept (Model-View- Controller)的特别的应用类 classes。从技术上来说，基本的类 classes 可能不是以那样浏览的，但是它的建造至少时逻辑化的，以使用 GUI 来建造应用。
- 一个基于 KDE 的应用框架窗口。这以应用类型为那些想从草稿开发自己的程序的人提供了很大的灵活性，但也可用作开发应用或模块的向导的基础。
- 一个只基于 Qt 的程序框架。对于那些希望使用 Qt-library 作为 GUI 界面的人，我们尽量为你提供一个聪明的框架以支持你的应用开发。由于它完全支持 Qt 编程，对于只使用 Qt 创建一个全功能的应用应该没有多大的问题。
- 一个 C++ 程序结构。这一应用类型旨在为那些希望编写基于终端的 C++ 程序的人服务。只需在 main()中去除"Hello World" 行并构造你的类 classes，方式上 Kdevelop 与 KDE 应用一样。
- 一个为 C 编程人员准备的 C 编程框架。这也是一个基于终端的应用，但只使用 C 编译器。

此外，Kdevelop 让你可以使用已存在的项目。它们可以由程序员自己通过设置和 Makefiles 来设定任何选项。至于执行及建立过程，当前状态只允许与其他基类相同的结构。用应用向导创建一个自定义项目并将你的文件加入项目以允许类浏览器的扫描。

为保证建立过程，你自定义的项目必须在一个匹配你的项目的小写字母名的子目录中包含所有资源；二进制文件的执行也限于此小写项目名。

注意，Kdevelop 并部把任何信息写入 Makefiles 或设置文件。必须由你负责任何项目操作并亲自进行设置。

### 8.1.2 库

创建库的一般项目类型暂时还不能用。无论如何，创建库对 KDevelop 来说不是不可能的。这里有一些方法指导:

- 无论何时当你的项目的子目录得到另一个包含资源文件的子目录,Kdevelop 会建立一个它们的静态库。那意味着静态库已获得自动创建的支持以挑选项目源文件。注意。静态库其后是二进制文件的一部分，不会得到安装。
- 要创建一个共享的库，你可以选择创建另一个项目子目录。在这个子目录中创建的源文件包含在项目内，因此在类浏览器中作为根部的类可用。要创建共享的库,编程手册 KDevelop 编程手册 提供一个 Makefile.am 模板。如果子目录的 Makefile 被加到 configure.in 脚本，你只需运行"Autoconf 和 automake" 及 "Configure"以创建 Makefiles。创建只有在子目录中使用创建命令才行，因为 Kdevelop 从原始项目子目录中激活创建。另一个创建共享库的可能的方法是根据编程手册里的模板，依照 项目改动 章节中中描述的项目修改规则，手动的改变原始项目录的 Makefile.am。
- 要安装一个共享的库，你必须查看 KDE-文件-系统标准，编程手册 KDevelop 编程手册中有描述。

### 8.1.3 多目标

对于一些项目来说，Kdevelop 当前状态的设备不会持久。那些是包含了多个目标的项目，好比包含了多个应用的包。象"执行"一样的命令要求开发者只建立一个目标。你必须编写你自己的到 Makefile.am 的入口并为创建另外的二进制和十进制文件创造目录，那些项目类型只有这样才获得支持。但是，一个建立过程

总是不顾及实际目标是什么而激活你的建立程序；这样这些函数仍然可以使用(限定建立是从主项目子目录激活的) 另一个仍然使用这种类型及得到二进制本身的方法是创建空的项目并将它们的子目录到其后会包含所有资源的目录并与项目文件相连 project files。然后你可以根据其项目文件独立的装载每个目标；这也允许执行及调试 debugging 目标。

按照 项目改动一章中解释的规则及以下的对编写主项目子目录的 Makefile.am 的指导(所有改动都在 KDevelop 写区域外)，主项目子目录内有多个二进制或库是可能的:

- 将你的目标加入 bin_PROGRAMS ，如果它可执行的话
- 加入你的库声明行，如果它是共享的库
- 加入与最初二进制的建立相同的声明:
    - newtarget_METASOURCES
    - newtarget_LD_FLAGS
    - DISTCLEANFILES
    - 复制信息: 原始二进制的入口，由 newtarget_SOURCES 代替 target_SOURCES, newtarget.pot 代替 target.pot
- 增加你的资源，例如 Kdevelop 二进制或库的写区域
- 要安装静态库, 用 Kdevelop 子目录内的"自动生成"来建造库。然后根据所需设置在书写区域外修改 Makefile.am。

## 8.2 新项目 New Projects

Kdevelop 应用向导允许四种不同类型的应用的创建，为每一个都创建一个框架。所有项目都使用 GNU 标准开发工具，如这本手册的要求部分所述。对于 KDE 应用，向导提供两种不同的框架，KDE-应用，提供一个带文件类型，视图及应用的基本类 classes 的完全的应用。这包括菜单条，工具条，状态条及继承了 KTMainWindow 类的主窗口的的创建。 KDE-Mini-应用只给出一个空的视图。这种类型的项目可用于创建小型桌面工具或其他在框架提供的已有代码上没有很多改变简单应用。一个 Qt-应用提供纯 Qt 程序的创建，如果你 wish no dependencies towards the KDE-libraries for end-users. Qt-应用也是通过使用树的基本类 classes，例如 KDE-标准-应用创建的，并提供菜单条，工具条及状态条。

C++应用类型提供一个框架以创建命令行应用。它准备好运行，缺省显示 "Hello World" 作为唯一操作。这对于那些希望先不使用 GUI 编写应用或那些使

用 C++或 C 进行命令行编程的的 C++ 学习者可能有用。C 编程人员也可以利用纯 C 项目，它要求任何 C-编译器最小化。

向导询问你的项目名称，版本及项目目录建立的地点。另外，你的名字及 Email 地址也会问及以将其插入头部和你的资源文件顶部的 cpp 模板，以及*.lsm 文件内的条目。

## 8.3 打开和关闭项目

Kdevelop 缺省设置为在启动时打开上次装载的项目。这允许快速启动，但你也许希望改变它以只启动 IDE 而不打开任何项目。要防止缺省行为，可使 KDevelop Setup 对话里的"装载上次项目"选项无效。

要打开另一个项目，从“项目”菜单中选择"打开"或按下工具条上的"打开项目"按钮。万一你当前已经打开了另外一个项目，它将被关闭。如果你当前的项目包含了未保存的文件，你将被询问保存文件。因此，你可以选择要保存的文件或关闭时不保存或一次全部保存。接着你会看见一个"大项目"对话，你可在此变为包含了要装载的项目文件的目录。Kdevelop 项目文件 project files 有*.kdevprj mime-类型，它也是由一个项目按钮显示。选择项目文件并按下"打开"。在装载项目时，类浏览器扫描所有文件并建立最初的类的树，这样你可以开始通过直接使用类浏览器着手于项目了。

另一个打开项目的好办法是在 KFM 中选择项目文件，KDE 文件管理器。这将启动 Kdevelop 并装载选中的项目文件。你还可以通过命令行打开项目，输入 kdevelop projectname.kdevprj。

在关闭 Kdevelop 时，你的项目文件会自动保存，IDE 会检测你是否改动过任何项目文件。然后会询问你在退出前是否保存任何改动过的文件。关闭时可用的选项和在装载另一个项目前关闭一个项目时使用的一样。

## 8.4 编辑一个项目

在你使用应用向导 KAppWizard 创建了一个信的项目后，通常的任务是通过编辑已创建的资源及加入类 classes，pixmaps，图片及其他任何你的项目需要的东西来扩展项目。现在，编辑一个项目意味着你可以在一个项目生成后根据你的需要，通过菜单及对话来改变缺省项目。因此下一节将叙述如何加入已有文件和类 classes 以及如何创建新的文件。在建立你的项目时需要这个，但这对你的终端用户的安装过程不会有所帮助。因此， 设置项目文件选项一节描述了如何设

置文件属性，尤其是你想要安装的其他文件，例如文档或 pixmaps。项目包含的另一个部分是为应用加入翻译以支持项目的国际化，在 加入翻译中有所叙述。

扩展项目文件 包含了关于如何创建一套好的使终端用户在遇到问题时能够进行自我帮助的在线帮助及如何使用你的产品的问题。最后，项目改动将描述你可以如何在特殊情况下进行 Kdevelop 项目管理。

## 增加/去除文件和类

增加新文件在你想分割你的类的实现 implementation 文档到多个文件时常常用到。然后你需要创建一个包含了你要去除的类的部分的实现 implementation 的新文件。你可以通过从文件菜单"File"-menu 中选择"新建"，打开"新文件 New File"对话来完成。这使你可以指定文件类型，名称及位置。当键入文件名时，Kdevelop 会自动为你输入扩展名，但你仍然可以根据你的参数选择改变扩展名。此外，你可以包含源文件的头部模板，这样你就无须亲自将它拷贝到你的新文件了。而且，你可以决定文件是否包含于项目中。注意，这并不包含安装目的地；它必须通过设置文件参数来设置。

在新文件创建以后，项目文件及相关的 Makefile.am 将更新。要把一个完整的类加入你的项目中，你可以从项目菜单中选择"新的类 New Class"，激活类生成器创建一个新的类。

万一你有一个已存在的项目并想加入某些想在你的项目再次使用的类 classes，可从项目菜单"Project"-menu 选择"将文件加入项目。。。"。选择"加入已有文件"，这会打开一个对话让你指定要加入项目的文件及目的目录。注意，文件将被拷贝到指定目录中并加入项目。你应该在增加资源后调用"Make"，然后你加入的文件将被包含到建立程序。万一你想加入，例如必须由终端用户安装的 pixmaps，你应该为增加的文件更新文件属性并指定安装路径(参见 Setting Project File Options)

要将一个文件加入已存在于项目目录内的项目，请使用 RFV，这里所有包含的项目文件都通过其项目状态在文件图标上有红色！显示。选择你想要加入项目的文件(它有一个标准的文件图标)并按下鼠标右键以显示弹出菜单。然后选择"加入"。文件也可从你的项目去除。在你不需要使用预先生成已由 KAppWizard 给定的文件时你可能会需要它。要去除一个文件，你可以选择从一个项目中去除文件或是完全删除。要从项目中去除一个文件，请在 LFV 或 RFV 中选择文件，按下鼠标右键并选择"去除"。要删除一个文件，请选择"物理删除"。

## 设置项目文件选项

文件属性对话可以通过项目目录或在 LFV 上鼠标右击进入。它以在 LFV 中的分组显示项目文件并显示文件属性，例如文件大小，文件类型，文件是否包含在项目中及安装路径--如果文件将由终端用户的安装命令安装。对于文档文件及 pixmaps 在项目创建并由终端用户安装时指定文件将去的位置很重要，因此你必须设置那些定位。对于标准 KDE 定位宏你应该查看你的 Makefile.am，它指定了你的定位宏。

## 加入翻译

由于 KDE 允许配置你的桌面及你的应用的行为，你还可以选择你的应用使用的语言，考虑到在线文档的使用及应用的外观。对于文档文件,这看起来象一个琐细的工作。你将加入一个由所希望使用的语言标注的子目录，例如德语的 de German，到你的项目的 docs 目录并拷贝英文文档到那个目录。然后你将生成整个文档并设置安装目录的所有项目文件的选项。接着你可开始翻译 SGML 文件到你想使用的语言并重新生成这个文档；这样你就完成了。对于应用，这对于程序员好象有点困难。现在，我们想解释如何使你的应用得到国际化支持及如何加入你想要支持的语言。首先，你必须由 i18n() 宏附上出现在你项目的控制条及对话中的可见字符串。这个宏是 Klocale 类的 klocale->translate()函数的替代，更易于使用。由于这个宏是在 kapp.h 文件中声明，你必须将#include <kapp.h> 加入资源文件或使用了宏的类的类声明 declaration 文件。还应该提到的是，虽然 i18n() 是一个宏，而你会考虑使用原始函数，但这行不通。因为设置来翻译的字符串必须从资源读出并存储在应用的翻译文件内 (<YourApp>.pot in the /posubdirectory)。这个任务是由程序 xgettext 完成的，要这样做，你必须在你的包含了这个资源的项目目录中键入 make messages。由于 xgettext 依靠 i18n()宏，原始的函数不能完成这个工作。

对于翻译本身，你首先必须使用 i18n()宏来创建包含了所有用于你的资源内的字符串的消息文件。这可以通过从建立菜单中选择"Make messages and merge" 来完成。然后你必须加入你的应用希望支持的语言。因此，从想菜单中选择"加入翻译文件"。这会打开一个语言选择对话。选好语言并按下确定。这将建造 ASCII 文件包含了文件名的条目 entries for the filename of the string and the line where the original string is placed. 然后你会看见一个包含了要翻译的字符串的 msgid 行,其后是 msgstr。Msgstr 行大多是空的,除了已由 KDE 库 KDE-libraries

提供的翻译。它们必须填充你的语言的相关翻译。

你可以考虑用手书写翻译，这样也可以。但是 KDE-SDK 提供用户 KTranslator 程序，它从安装在系统上的其他程序读出已有文件，这样你可以重新使用已翻译的字符串以支持你的语言。

要进入 KTranslator，最简单的方法是在 LFV 或 RFV 内选择一个/po 目录内的 <language>.po 文件。这会打开 KTranslator 并允许轻易的完成翻译。注意，你必须由你自己设置 KTranslator 属性以包含作者名字，语言及目的文件。缺省情况下 KTranslator 只打开你的翻译文件。对于所有有的"文件，make 使用 msgfmt 程序来格式化你的信息文件以使用二进制，但是你无须操心这一点以及指定安装翻译文件的目标目录；这是由 Kdevelop 自动完成。

要得到关于国际化支持的详细信息，请参见 http://www.kde.org；很多人在为你翻译以支持他们的语言。你会发现一个翻译人员的 email 地址列表，你可以写信给他们，他们可以帮助你。还可以阅读 对话框编辑器及 Kdevelop 编程手册，在这里再次谈到了关于国际化的问题。

## 扩展项目文件

所有 Kdevelop 创建的项目都包含了一个预先设置的文件，它已经包含了安装的标准章节，项目名称，版本以及作者名和 email 地址。由于 Kdevelop 使用的是 SGML-模板，把文件扩展为完整描述的帮助系统十分容易。你唯一必须完成的事是编辑 SGML 文件，它作为 index.sgml 放于 docs/en 中。你的 sgml-工具包包含的参考文件可以加入帮助浏览器 Helpbrowser，允许你直接进入特殊标签及关于如何扩展文档的简短描述。SGML 有很多优点，而 KDE 利用其他的 KSgml2Html 工具通过文件类型广泛使用。这会创建典型的 KDE-风格的文件并让它看上去更好。不管怎么说，sgml 工具本身就已经足够产生一个已包含于你的项目的 html 输出 output。要使用 KSgml2Html 创建文件，先安装工具并从建立菜单运行"生成用户手册"。文件浏览器允许通过从帮助菜单或文件树 DOC-tree 的相关按钮选择"项目用户手册"来直接控制输出 output。然后你可以在 KDevelop 中浏览在线文件，更好的通过输出 output 浏览导致标签丢失的错误 errors。

现在，在展开文档时，你无可避免的会生成一些必须包含于项目的其他文件，由于每个 sect-tag 都创建一个新的 HTML 文件。由应用向导 KAppWizard 生成的输出 output 已包含于项目文件，因此你无须操心它们的安装路径。你真正需要关心的是任何的 index-xx.html 文件，xx 是大于 6 的数字(由于前 6 页已经包含在项目中)。在生成了文档以后，切换到 RFV 并浏览你的文件目录。在文件上按下

鼠标右键并选择"增加"。另外， KSgml2Html 将 KDE logo 加入文件目录。这个文件，logotp3.gif，也必须加入项目。然后你必须从项目菜单或通过文件浏览器的弹出菜单选择"文件属性"。设置安装路径最简单的办法是选择一个已经设置好安装的文档文件，例如 index.html。你可以看见 Install 已监测安装目录及文件名也已包含了目的地。标注安装目录并键入 CTRL+C 以拷贝安装路径到剪贴板。接着选择你指定要安装的文件。激活 Install，这将激活安装目录入口域，已包含了文件名。将光标置于文件名之前并键入 CTRL+V 以插入剪贴板上的内容 (这是起先拷贝的安装路径)。这是指定安装路径最快的方法。要得到关于指定安装目的地的更多选项，请参见 KDevelop 编程手册。

## 项目改动

当着手于一个项目时，你决不应该手动的编辑项目文件。在一定情况下这会阻止 Kdevelop 正确装载你的项目，而且改变也不会使 Makefiles 更新。要改变你的项目的任何设置，你必须使用给定的工具条目，例如，增加文件或设置文件属性的条目。对于不习惯某些设置的专家，例如连接器 Linker 或需要其他项目设置，你应该研究一下 Makefile.am 的宏并在以"Kdevelop 写区域"条目分割的部分之后将所有改动加入 Makefile.am's。由于 GNU-工具是在所有宏文件的末尾使用这些命令，你可以用它覆写 Kdevelop 的设置。注意，这会阻止使用 KDevelop 进行与项目设置相关的任何改动。

# 8.5 项目的编译和连接选项

每一个新项目都包含了编译 Compiler 和连接 Linker 需要的所有选项及一般设置。缺省情况下，你的项目设置为通过-g 标志使用编译 debugging，警告 warnings 也设置为标准型，-Wall。这保证你可以调试你的应用并监测可能导致程序错误 errors 的建造。对于有些应用。你会需要其他的编译器 Compiler 或连接 Linker 标志，尤其是当你使用当前并不包含于连接器 Linker 的库时。然后你需要通过使用项目选项对话来正确配置以更新项目。参见 建立设置以获得更多关于如何及在何处设置编译器选项，警告及连接器 Linker 选项。

# 8.6 外部项目

已有项目可通过从项目菜单中选择"新建"转换为 KDevelop 项目。以下对话

用你的项目名，版本，类型信息以及你的名字和 email 地址创建了一个空的项目文件。然后拷贝你所有的编译及建造文件到新项目的目录并从项目菜单中选择"增加文件"。选中的文件将被拷贝到你的项目目录中，Makefile.am 也会更新。请改变所有转变前存在的 Makefile.am 入口为由 Kdevelop 在 Kdevelop 区域新建的入口。测试一下在转换后你的项目是否仍然可以编译及安装以保证项目的稳定性。

# 9. 建立设置

项目选项对话可通过项目菜单进入，它允许你指定所有你的项目需要的参数。它们将用于 Makefile.am's 及 configure.in 脚本(例如，版本号改变及编译器警告 compiler warnings )，因此也设置编译参数。在改变了项目选项后，你应该激活"清除 make clean" 或"全部重建"以使用新选项来编译你的项目。请注意，调试 debugging 只有在设置了项目选项以生成调试 debugging 信息时才可用，可通过调试等级 (0-3)设置数量。如果你在连接器 Linker 标志中加入了属于未包含的库的信息，而它们都未更新，则你的程序不会正常连接，因此应该记录你的二进制连接。

为生成你的应用的发布或为发布源代码包，你应该检查以下标准设置：

- 使编译 debugging 失效
- 允许优化并设置优化级别为**-02**
- 设置编译器警告 compiler warnings 为**-Wall**
- 对每个新发布的软件，为了版本和需求升级版本号并更新项目文件

## 9.1 一般选项

项目选项对话的第一页为你的项目设置一般选项。即项目名号码，手册中用于生成一套包含于项目的 HTML 文件的 sgml 文件及关于作者的具体信息。简短描述区域可输入你想要包括的其他信息，如程序的目的等。

## 9.2 编译器选项

编译器选项 compiler options 页给你的目标设置编译器标志，调试 debugging 及其他。

### 目标

目标格包含了三个可设置的选项:

目标机器: 你可以在此通过在你的机器（缺省）和 i386v 之间作出选择来设置目标选项,如果你把你的编译器 Compiler 设置 configured 为一个运行 V 系统的

Intel 386-兼容机器的交叉编译器，可选择此项。这一选项把-b 标志设置到编译器 Compiler。通常，这一项保留缺省值。

只做语法检查：选中后，会设置只做语法检查标志。这意味着编译器 Compiler 将只检查你的代码的语法正误，但并不检查此外的其他项目。

优化：你可以通过这个选项允许优化建立过程，意味着设置-O 标志。如果未选中，标志将设为-O0，这样就不会使用任何优化。如果你通过选择此选项允许优化，你也可以在其下从 1 到 3 指定优化级别。

对于你的应用的发布版，应允许优化并将标准设置为 2。

## 调试

在目标方框的右方，你可以看见调试 debugging-选区。这意味着你可以设置你的编译器 Compiler 以在最终二进制内包含调试信息，这样程序员就可以跟随应用的执行，并随调试器直接看到资源代码上下文。

因此激活调试 debugging 设置 -g 标志；调试 debugging 标准指定包含于二进制的信息量。可选择 1 到 3 级。注意二进制的执行可以通过设置任何调试 debugging 选项减慢，二进制文件大小则通过调试 debugging 标准增大。

生成 gprof 的额外信息：设置-pg 标志，这将导致编译器 Compiler 包含展示你程序功能的调用图表的 gprof 程序的信息。

存储临时交互文件：设置-保存-临时文件 标志。这将导致存储通常由预处理器及集成器产生的的临时文件。因此，源文件的编译将产生三个输出 output 文件：一个 *.o 文件，它是编译器 Compiler 的最终输出 output，一个由预处理器产生的*.i 文件及一个作为集成器输出 output 的*.s 文件。

要发布你的项目，请使任何调试 debugging 可用。

## 其他选项

底部的文本条目区域旨在通过在 Makefiles 里设置 CXXFLAGS 环境 variable 来手动的设置任何编译器 Compiler 标志，因此应该在建立过程之前设置标志并在其后对它们重启。要得到所有可用的编译器 Compiler 标志的完整描述请参见你的编译器 Compiler 文档；对于 gcc 和 egcs，这可由 **man gcc** 来完成；**man g++** 会告诉你关于用于指挥编译 Compiler 的 c++脚本的信息。

# 9.3 编译器警告

以下将描述关于可以在项目选项对话第三页进行设置的编译器 Compiler 警告选项。说明取自 GCC 页, egcs1.1.1.版。警告 warnings 本身是诊断信息，表明创建可能引起错误 errors。

**-Wall**

综合的标准的`-W'选项。

**-W**

用-W 编译。这已选项设置了不时很特殊的未包含在-Wall 中的选项。请阅读 GCC-Info 以获得更多信息。

**-Wtraditional**

警告在传统或标准 ANSI C 中表现不同的某种构造。

**-Wundef**

在一个未定义标识在一个`#if'指令中作出评价时警告。

**-Wshadow**

无论何时当本地变量 variable 隐蔽了另一个本地变量 variable 时警告。

**-Wid-clash-LEN**

无论何时当两个不同标识符在第一个 len 字符匹配时警告。这可以帮助你准备一个将由一定编译器 Compilers 程序编译的程序。

**-Wlarger-then-LEN**

无论何时当一个比 LEN 字节长的对象被定义时警告。

**-Wpointer-arith**

警告依靠一个函数类型或空的大小的任何事物。GNU C 给这些类型大小赋值为 1，为方便计算使用 void * 空指针及函数指针。

**-Wbad-function-cast**

无论何时当一个函数调用是用于一个不匹配类型时警告。例如，当 int malloc() 用于 anything *.时警告。

**-Wcast-equal**

无论何时当投出一个指针以从目标类型去除一个类型限定时警告。例如，在一个 const char *投向一个普通的 char *时警告。

**-Wcast-align**

无论何时当投出一个指针以使目标所要求的队列增加时警告。例如，当一个 char * 投向一个在整型只有在 2 或 4 个字节限定时才能访问的机器上的 int *时警告。

**-Wwrite-strings**

赋予字符串常量予 char[length]常量类型，这样拷贝其一的地址到一个非常量的字符*指针时会得到警告。这些警告 warnings 将帮助你找到一个编译时间代码，它可以尽力写入一个字符串常量，但只有当你在声明 declarations 及原型中使用常量时十分仔细才可。否则，它就只是一个损害；这就是我们没有让`-Wall'请求这些警告 warnings 的原因。

**-Wconversion**

当一个原型引起一个不同于没有原型时发生于同一 argument 的类型转换时警告。这包含定点转换为浮点，反之亦然，及改变定点数范围和有无符号的转换，除了和缺省的晋级一样时。

**-Wsign-compare**

当有符号数转变为无符号数，而有符号和无符号的值的对比将产生一个不正确的结果时警告。

**-Waggregate-return**

当返回结构或联合的任何函数被定义或调用时警告。(在可以返回队列的语言中，这也会引起警告。)

**-Wmissing-prototypes**

如果一个全局函数定义却没有前面的原型声明 declaration 时警告。即使定义本身提纲了原型这个警告也将发出。其目的是监测头文件中未声明的全局函数。

**-Wmissing-declarations**

当全局函数没有先声明 declaration 就定义时警告，即使定义本身提供了原型。使用这一选项来监测头文件中未声明的全局函数。

**-Wredundant-decls**

当某种事物在同一范围内的声明了超过一次时警告，即使是在多声明 declaration 有效且没有改动的地方。

**-Wnested-externs**

如果一个外部声明 declaration 在一个函数中遇到时警告。

**-Winline**

如果一个函数不能内联，而它要么声明为内部，要么给出 -fin*line-functions 选项时警告。

**-Wold-style-cast**

如果一个老风格(C-风格)调用用于次序时警告。

**-Woverloaded-virtual**

(C++ only.) 在一个派生类中，虚拟函数的定义必须匹配在基本类中声明的虚拟类的签名类型。当一个派生类声明一个可能是一个定义虚拟函数的错误尝试的函数时，使用此选项来请求警告 warnings；也就是说，有和一个基本类中虚拟函数同名的函数，但有一个不匹配任何基类中虚拟函数的类型签名时警告。

**-Wsynth**

当 g++综合行为不匹配 c 前端时警告。

**make all warnings into errors**

(-Werror) 把警告 warnings 看作错误 errors；在任何警告后退出编译。

要发布你的项目，推荐使-Wall 可用。

## 9.4 连接器 Linker 选项

连接器选项 你当前项目的连接器 Linker 选项可以通过项目选项对话的最后一页设置。你必须激活那些你的应用中用于利用连接器 Linker 将它们连接到二进制文件的库，例如，你的应用使用类文件对话 KFileDialog。由于类文件对话 KFileDialog 是 Kfile 库的一部分，你必须激活 kfile。对于 checkboxes 中未列出的类 classes 或函数，可使用其他库 **"additional libraries"**域。

### 连接器标志 Linker Flags

连接器标志

**remove all symbol table and relocation information from the executable:**

这意味着所有多余的信息都将从项目文件及二进制文件中去除，并导致无法调试 debugging。只要你的应用是处于开发阶段，不是作为最终的发布，你应该让这个选项无效。

阻止使用共享库

**prevent using shared libraries:**

这一选项使使用在支持它的机器内的共享库无效。在不使用共享库的系统上，这一选项无效。

其他标志

**additional flags:**

在此，你可以为连接器 Linker 键入另外的标志，通过 make 设置 LDFLAGS 环境变量 variable。变量选项可以从 ld 页或你的编译器 Compiler 页中得到。

# 库

库这节包含最需要的库的 checkboxes 连同 Qt/KDE 应用开发。你必须激活那些你的应用要使用的库，否则连接器 Linker 会抱怨未定义的代号表格。

**X11**

The X11 library. 对于所有 X-Window 程序推荐使用。

**Xext**

The X11 extension library. 同样大多数 X-Window 程序要依靠 Xext。

**qt**

The Qt-library. 推荐 Qt 及 KDE 应用使用。

**kdecore**

KDE 核心库，包含类 KDE 应用框架的 classes。

**kdeui**

KDE 用户界面库；包含 KDE-特殊控件 widgets。

**khtmlw**

KHTML 控件库。

**kfm**

KFM 库，包含 KFM 函数的类 classes。

**kfile**

KFile 库；包含文件对话等。

**kspell**

Kspell 库；包含让程序为检查拼写使用 Ispell 的界面。

**kab**

KadressBook 库。访问地址薄及提供地址薄控件 widgets 时需要。

**additional libraries:** 在此你可以输入你的应用需要的其他的库；例如 KOM 库。用 -l 选项设置库，例如-lkom。

# Make

由于 GNU make 支持一些有用的选项，项目选项对话包含了一个叫做 "Make-Options"的页面，在此它们可以设置为可用/不可用。可用设置为：

打印调试信息

**Print debug information**

打印出所有关于项目文件 project files 的信息和 make 重建它们时的测定。错误后继续

**Continue after errors**

尽量在错误发生后继续编译(例如一个文件由于某个错误不能编译) 打印数据库

**Print the data base**

打印出包含了上次运行建立后的当前过程的数据库 make-database。环境变量

**Environment variables**

给当前环境变量一个比当前在 Makefiles 中使用的变量更高的优先级。没有内置规则

**No built-in rules**

不使用 make 的内置程序。

**Touch files**

不在改动的文件上运行编译器 Compiler；而只接触它们。这将它们设置为已由 make 处理。忽略所有错误

**Ignore all errors**

忽略发生的所有错误 errors 静态操作

**Silent operation**

不打印处关于建立过程的任何信息打印工作目录

**Print working directory**

打印 make-过程中的当前目录工作成员

**job number**

为 make 设置相似程序量。对于一个单 CPU 系统，我们推荐将此设置为 1 或 2 设置修改

**set modified**

设置选中的修改文件。通过单击右边的文件夹按钮选择文件。将一个文件设置为改动意味着该文件将由 make 处理和编译，如果它是一个资源文件。其他选项

**additional options**

设置其他选项到 make；它们可以在你本地发页面中为 "GNU Make"找到。

## 9.5 Make 选项

正如 GNU make 支持许多有用的选项一样,在项目选项对话框中有一页叫"连编选项", 你可以选择要使用的选项.可用的设置有:

**打印调试信息**

打印关于项目文件以及为什么重建的所有信息.

**发生错误后继续**

在错误发生后(比如某文件由于错误无法成功编译)将试图继续编译.

**打印数据库**

打印当前进程自上次建立以来变化的数据库.

**环境变量**

让环境变量值优先于 Makefiles 中的变量.

**无内建规则**

不使用 make 的内建规则.

**更新文件修改日期**

不编译改变的文件;只是改变其修改日期.

**忽略所有错误 errors**

忽略所有的发生错误 errors.

**安静操作**

不打印在编译连接过程中的任何信息.

**打印工作目录**

在 make 过程中打印出工作目录.

### 任务数

设置并发 make 进程的总数.对一个单处理器系统,我们推荐将起设为 1 或 2.

### 设为已更改

把选择的文件设置成已更改过.按右边文件夹按钮以选择文件.设置一个文件为已更改意味着,如果它是源文件将由 make 强制处理和编译.

### 设置附加选项

为 make 设置附加的选项;那些选项可以在你本地的 man page 中找到.

### 运行时的重建行为

设置你在选择运行时发生什么动作.你可以有三种选择:

- 总是重建: 不检查更改的文件而总是重建.
- 更改时警告: 如果源文件已经更改则发出警告并询问是否重建项目.
- 仅在更改时: 检查是否有源文件已更改并重建更改过的文件.

缺省的是总是重建.

# 10. 类浏览器

## 10.1 类浏览器

类浏览器 Kdevelop 的类浏览器是 IDE 提供给开发者管理他的项目资源最有用和最重要的工具之一。当一个项目装载后，一个类语法分析器将读出函数，类 classes 等的所有项目资源，然后加工结果显示在 CV 树视图中。本章为你展示如何使用类浏览器和它提供的功能及它是如何改善你的工作的。类及其方法也可通过浏览器工具条访问。在那里，左边的 combo 选择类，右边的让你选择选中的类的方法。当你选择了一个方法，类浏览器会自动把你带到实现 implementation 文件并将光标置于方法中。最后，点击方法 combo 右边的类辅助按钮后它会带你到方法的声明 declaration 处；再单击定义。随后的弹出菜单---由按钮上的向下箭头显示---会提供其他的在类浏览器上下文菜单中可用的功能，例如：

- 到声明处：浏览方法的声明 declaration
- 到定义处：浏览方法的定义
- 到类声明处：浏览类的声明 declaration
- 新的类 New Class：打开新类 New Class 对话以构造新类
- 加入方法：增加一个方法到选中的类
- 增加属性：增加一个属性到选中的类

### 10.1.1 可用对象

通过可用对象，我们描述了一个术语，意味着 C++代码可以看作一个对象类 classes 的集合，它们的成员，全局函数等。类树逻辑的显示这些对象并通过特征排序，因此它们在树中很容易定位。所以类书包含了一个"Classes" 和一个"Globals"文件夹。这样"Classes"文件夹一般包含项目的类 classes；如果你的包含了子文件夹以管理你的资源文件，它们也通过其原始文件夹名称显示并包含存储在为于子文件夹的文件中的所有类 classes。

此外，当弹出一个类，类树将通过分隔方法和属性 attributes 显示类的内容。由于这些也可以有属性 attributes 如公有，私有和保护，它们也可以通过标识过的按钮显示。你可以看见类浏览器中显示的一个类包含了出现在类声明 declaration 中的所有对象。

现在，在用 C++编程时，类 classes 是一个很普通的事物，也将包含大多数代码。但是应用也包含在项目中有"全局"外观的对象。这些可能时结构体或函数等。尤其 main()函数在每个应用中都出现，有时你需要以某种方式修改它。要访问这些对象，类浏览器提供了" 全局" 文件夹，包含了以下对象类型的子文件夹：

- 结构体
- 函数
- 变量

由于显示这些项目的按钮与那些用于类浏览器的相似，它们的意思很容易被程序人员猜到和记住。

最后，可以说类浏览器 classviewer 是通过在代码中与其外观相关的对象用图表法显示你的项目。以下将连同你的代码教你如何使用类浏览器 classviewer 及其工具。

## 10.1.2 浏览对象声明及实现

类浏览器最强的功能是通过代码的上下文提供其代码的快速访问而不管在文件中的位置。因此，鼠标点击选取将导致以下操作：

- 类名 **On a classname:** 切换到类声明 declaration 处
- 类方法 **On a class method:** 切换到方法实现 implementation 处
- 类属性 **On a class attribute:** 切换到类声明 declaration 中的属性声明 declaration 处
- 结构体 **On a struct:** 切换到结构体声明 declaration 处
- 全局函数 **On a global function:** 切换到函数的实现 implementation 处

现在，这提供给你对代码对象最需要的访问。显然，它也许需要改变一个方法的头部，这导致你必须在类中改变它的声明 declaration 及实现 implementation。类浏览器 classviewer 通过提供在项目上右击后的出现的上下文菜单来支持它。对于一个方法或函数，这意味着你可以选择要去的地方：

- 到定义处 **Go to definition:** 切换到实现 implementation 处--这是左击时的缺省，如上所述。
- 到声明处 **Go to declaration:** 切换到方法或函数的声明 declaration 处。

通过这一性能，类浏览器让你可以访问你为编写 C++应用必须去的每个地方。

以下将描述类浏览器提供的其他工具，你将发现它们在着手大项目的时候十分有用，因为它们推崇面向对象的 C++编程工作。

## 10.2 类工具

类工具 类工具是让开发者更起义的获得关于他项目的类 classes 的更多信息的对话。类浏览器 classviewer 通过项目在代码中的出现显示所有项目，但是你有时希望无须深入查看代码而得到关于类 classes 的更多信息。因此，类工具对话专门用于显示特定的类属性 attributes。

类工具对话是通过类浏览器 classviewer 中的类上的弹出菜单激活的。选择 ″类工具″，对将话出现。要得到某个类的通知，在顶部的 combo 框中选择这个类。然后工具条中的按钮将为你提供你的类的特殊的树的函数。它们是：

双亲

**Parents:**

选中的类的双亲，意味着它所继承的类。对于多继承及查看为什么一个的行为很有用，例如，对于对话你的双亲类将是 QWidget 或 QDialog。孩子

**Children:**

显示继承当前类的孩子类 classes。客户

**Clients:**

通过类声明 declaration 中的属性使用选中类的类。供应方

**Suppliers:**

为选中类提供属性 attributes 的供应方。属性

**Attributes:**

通过其名显示的类的属性 attributes。方法

**Methods:**

选中类的方法。虚拟方法

**Virtual Methods:**

一个类提供的虚拟方法。

此外，选择的属性--公有，保护，私有或全部--通过其属性值显示其属性，方法及虚拟方法。

## 10.3 管理类

另外类浏览器允许通过对话直接加入方法和属性 attributes。这意味着你无须自己打出类声明 classdeclaration 及实现 implementation 的头部。在加入一个方法后，你只需设置实现 implementation 头部的形式参数，如果方法需要一个属性，加入声明 declaration。

**» 如何为类增加方法**

1. 选择你想要增加方法的类
2. 按下鼠标右键，出现弹出式菜单
3. 选择"增加成员功能"
4. 显示"增加成员功能"对话
5. 插入方法的类型，申明 declaration 及文件
6. 指定方法的通路就修饰字
7. 按下确定以退出对话

要加入一个变量 variable，操作是一样的，只要在弹出菜单中选择"增加变量 variable 成员"。

这些对话操作的区别是增加一个变量 variable 将把变量 variable 增加到类声明 classdeclaration 中,增加一个方法将把方法的声明 declaration 及方法的实现 implementation 头部增加到资源中。由于类浏览器 classviewer 直接自我更新，你可以直接进入新方法的实现 implementation，因此你只需填写方法的实际目的的代码。

# 11. 对话框编辑器

Kdevelop 集成的对话框编辑器 dialog editor 允许通过图表的方式对控件 widgets 和你的应用使用的对话进行简易创建。你可以直接看见你的对话的外观，就象它为用户显示的一样。使用对话框编辑器 dialog editor 通常是你在创建了一个新的项目并用应用向导 KAppWizard 创建你的主视，用户交互对话，和在完成图解工作及代码生成后要完成的第一步。这样，你的项目将包含所有通常认为是很"困难"、一般要花很长时间才能实现的部分。然后，你剩下的工作就是实现生成的代码中的功能了。本章涉及如何使用对话框编辑器 dialog editor 创建你的项目控件 widgets 及当你觉得你的控件 widgets 在进一步的开发过程中需要更正或增加时应该如何做。

你可以通过从视图菜单"View"中选择"对话框编辑器"或通过工具条相关按钮来切换到对话编辑器 dialog editor。要切换回项目编辑器，可从对话框编辑器 dialog editor 的视图菜单中选择"KDevelop""View"或按下相关工具条按钮。

对话框编辑器 dialog editor 的界面还为你提供了什么呢?主要来说，按照主视图的分割，它的外貌和项目编辑器以及菜单条，工具条几乎一样。这允许你可以很快的习惯对话编辑器 dialog editor，而且，由于他完全和项目管理进行交互,如果你想要控制你的建立过程，你可以一直在对话框编辑器 dialog editor 里。需要切换回 KDevelop 的操作将为你自动完成它，比如进入文件浏览器。只需选择菜单命令，Kdevelop 将如你所愿。

下面的章节总体的介绍了对话框编辑器 dialog editor 界面，最初如何创建一个新的对话及如何设置你的对话包含的孩子控件的 widgets 属性。

## 11.1 对话框编辑器视图

### 11.1.1 主视

对话框编辑器 dialog editor 视图逻辑分割为：

- 控件定位装置，包含"控件","对话" 及"项目"制表符。参见 添加控件中的描述。
- 控件编辑器，代表创建你的对话的编辑视图，参见 控件编辑器。

- 属性窗口，包含属性及其值的列表，由当前再控件编辑器中选中的的控件决定。参见 设置属性以获得关于如何指明控件行为及外观的信息。

与 Kdevelop 不同的菜单条 Menubar，工具条 Toolbar 及状态条 Statusbar。

在对话框编辑器模式，Kdevelop 对菜单条，工具条及状态条有些许改变以提供你创建控件 widgets 所需要的功能。它们是:

### 11.1.2 菜单条 Menubar

文件菜单**"File"-menu:** 用"新的对话"取代"新的"。"打开"允许打开一个对话定义文件。

视图菜单**"View"-menu:** 用"控件-视图"取代"树-视图",激活/不激活控件-视图定位装置；加入"属性-视图"以激活/不激活属性-视图和"网格尺寸"来让你用水平及垂直的像素值指定网格尺寸。

建立菜单**"Build"-menu:** 由"生成资源"代替"编译文件 Compile File"。这让你实际生成你对话的资源。

### 11.1.3 工具条 Toolbar

工具条包含一个"新的对话"的新按钮，并由"生成资源"代替"编译文件 Compile File"。

### 11.1.4 状态条 Statusbar

状态条为你提供关于当前选中的控件的信息，尤其在坐标系中显示 X 及 Y 的值。对于状态条 Statusbar 帮助，你将被提供与项目编辑模式同样的功能。

在改变控件尺寸时，状态条会显示选中控件长宽的当前值。

## 11.2 创建新的对话

在创建了你的项目骨架后，按照你的参数选择将为你提供一个运行准备好的了应用。由于 Kdevelop 提供项目类型 KDE 及 Qt 应用，对话框编辑器 dialog editor 认可它并由以前的库提供的控件 widgets 来提供控件构造。要节省时间，你还需在头脑中由格设计来完成想要的操作。要得到关于控件设计的信息，请参见 KDevelop 编程手册。要创建一个新的对话，请从文件菜单"File"-menu 中选择"新

建"或在"对话"标签上从上下文菜单中选择"新建"。"新的对话"菜单会出现，你必须在此给 Kdevelop 一个对话---关于基础类资源文件名称及目的地的具体信息。

## 11.2.1 对话类

你可以选择的对话类，是由你的新控件继承的类，专门由类本身来代表。因此，你有以下选项：

1. **QWidget:** Qt 提供的所有用户交互控件 widgets 的基类。用于主视及最高层的控件 widgets。
2. **QFrame:** 继承 QWidget 并由众多控件 widgets 用作基类。这对于除 QWidget 方法外还想拥有 QFrame 功能的控件 widgets 很有用。
3. **Custom:** 继承一个必须在"自定义属性"中设置的自定义类。这可能是一个由你的项目提供或库的已经设计好的类。
4. **QDialog:** 你要继承为用户交互--例如设置属性或改变参数--使用的对话的基类。
5. **QTabDialog:** 继承 QDialog 并提供一个有预先定义的按钮及一套标签的对话，这是你将由你要创建的控件 widgets 提供的。

自定义属性 Custom Properties

对在对话类域中选中的一个自定义类的继承，你必须指定类名，在"自定义类"里。对话编辑器 dialog editor 使用它来生成代码；因此，你还必须插入"自定义头部"，在此还必须设置自定义类的头文件名。

## 11.2.2 文件

在"文件"章节，你必须键入控件的具体信息。即类名(这将是，例如对于一个允许选择钢笔颜色的对话：KColorSelectDlg)，头部，C++ 及数据文件名。当插入类名，文件名将由对话编辑器 dialog editor 指定，但是你也可以改变文件名。现在，你的文件又如何呢？当你准备构建可视化控件时，你将不得不生成包含了你的控件的实现 implementation 的文件。由于这将是一个类，对话将与包含了类声明 classdeclaration 的头文件，一个包含了你的控件的方法和槽的实现 implementation 方法的 C++文件一起存在。数据文件是包含了一个可由你的控件的构造器及 *initDialog()* 方法调用的函数。由于它将包含从对话框编辑器 dialog

editor 生成的代码以在屏幕上创建控件这个文件本身不应该改动。如果你必须改变参数值，你应该通过构造器来完成或保证你在开发过程中不会改变对话，因为每次生成可你的控件的代码，数据文件都会被重写。头部及 C++ 文件包含多个零件，对话框编辑器 dialog editor 会在此编写;这些由评论标注。在文件生成后，你可以改变任何值并在这些零件之外设置； 否则你的改动会在下一次代码生成时丢失。

### 11.2.3 定位

要生成控件的资源，对话框编辑器 dialog editor 需要知道他们将被设置的位置。 output 目录的缺省值是当前项目的子目录，它包含已有的资源。

在按下"确定"后，你的缺省值就已经生成，一个空的控件构造器也打开。然后你就准备开始创建你自己的控件了。注意，对话框编辑器 dialog editor 当前只支持没有几何管理的静态控件 widgets 。如果你准备为你的控件 widgets 使用几何管理，你应该让自己习惯于 Qt 为此提供的类 classes，使用类生成器创建一个新的类并自动编写自己的控件。要获得更多信息，请参见 KDevelop 编程手册。

## 11.3 增加控件

在指定了对话或控件 widgets 的类及文件名以后，你可以准备开始创建控件并充实其内容.为你的对话增加低水平的控件 widgets 是件很容易的事. 只需从左边的"控件"定位装置中通过在相关控件按钮上单击选择你要加入的控件. 然后控件就会放于当前打开的主控件的左上角. 一个增加的控件会在编辑器 editor 视图中得到缺省的 100x30 像素的大小. 要移动一个控件，在其上点击以激活绘图框架，它以深灰色显示，控件的顶部，底部，左边，右边的中间及角落上都有热点。十字光标表明控件可以移动。要移动它，请按下鼠标左键不动，然后用你的鼠标移动控件到你希望它其后显示的位置.

要对一个控件重新设置大小，请在一个已经激活的项目的热点上方移动你的鼠标光标。鼠标光标会变为双箭头，表明重设大小的方向。按下鼠标左键不放。当鼠标移动到光标指示的方向时，控件项目会改变它的大小。

另外，控件编辑器包含许多上下文菜单以帮助你协调你的工作。这些对控件 widgets 定位器中所有项目都可用，并提供一个快速帮助信息框，显示选中控件的类名并给予简短叙述。在选中控件上方，上下文菜单显示选中项目的类名并提

供:

- 升高
- 降低
- 升到顶部
- 降到底部
- 剪切
- 删除
- 拷贝
- 粘贴
- 帮助

在设置了尺寸和位置后，你可以在参数窗口编辑选中项目的参数。

### 11.3.1 控件定位装置

对话框编辑器控件 控件 widgets 标签代表你可以放于对话的控件 widgets。如果你想得到关于某个控件的信息，请在控件按钮上按下鼠标右键并从弹出菜单中选择"快速-帮助"。注意如果你的项目类型是纯 Qt 或 KDE 的，对话框编辑器 dialog editor 是自动测定的。这会阻止你在 Qt 应用中使用 KDE-widgets。在你选中了一个控件项目后，它将以缺省尺寸及参数值放置于编辑器窗口并由一个框架和涂黑的角标注为选中. 要改变一个控件的大小，请在一个黑色的点山移动你的鼠标，你的光标会改变以显示可能的变改方向. 然后一直按下鼠标键并移动鼠标. 当你完成控件尺寸的改动后,松开鼠标. 在改变尺寸时，状态条将通过 X 和 Y 的值显示项目的当前位置，并以 W(宽) 和 H(高)的值显示当前大小.

### 11.3.2 对话定位装置

项目对话 对话定位装置旨在让你通过鼠标点击打开你的项目的对话. 由于对话的结构时保存于在包含了生成的文件的目录中的一个*.kdevdlg 文件内，只有那些对话定义文件会显示. 还要注意不要删除了这些定义文件.

选择了一个对话后，它会在控件编辑视图中显示上一次编辑步骤后保存的状态.

### 11.3.3 项目定位装置

对话项目 项目定位装置让你可以分级的浏览当前显示的对话的控件项目. 这意味着, 由于你的背景代表了对话中所有控件 widgets 的父体, 它将显示在树的顶部. 主对话的孩子则列于树的下一个级别.

选择了一个项目后, 它会标注于编辑器视图中, 其属性也会显示于属性窗口. 使用项目视图有时很重要，如果你控件 widgets 的行为要依靠父子关系的话。

## 11.4 控件编辑器

控件编辑器是居中的主视，你棵在此构造你的控件. 在增加了一个控件后, 它们可以选中并重设大小, 或移动到你需要的位置. 对于所有项目, 弹出菜单会提供快速进入例如剪切，复制，插入等操作的途径。

## 11.5 设置属性

设置控件属性 右端的属性窗口是你可以设置控件及其项目缺省行为的地方. 它会立即展示每个选中项目的预设值; 改变参数值会导致控件编辑器视图的直接变化, 例如，名称标签或按钮.

要通过其效果分开一定的属性值, 属性窗口包含四个文件夹; 选择其一会弹出属性组的所有值. 所有可能的值都得到描述. 注意，属性相对于控件独立, 例如，一个标签和按钮会有在显示屏幕上的名称属性, 而行编辑会有方法属性，例如 *setText()*.

对于每个项目可用值的完全列表, 你可以查看控件的类-参考,它解释了所使用的方法及所有可能的值. 注意，大多数值在 QWidget 中得到应用，并由所有继承了 QWidget 的 widgets 所使用. 还要注意最终代码并不包含任何不可由用户改变因此只能使用控件构造器所给定的缺省值的方法调用.

所支持的可在属性窗口为每个控件项目设置的属性的完整列表.

## 11.6 生成文件

对话源代码的生成

在生成了一个控件后, 你必须生成源代码以让它在你的项目中可用. 这可以

由"建立"菜单的"生成资源"或通过对话框编辑器 dialog editor 工具条上的相关按钮来完成. 你的 Makefiles 将自动更新以在编译程序中包含新的控件; 因此, 在调用了"生成资源"以后, 你可以在对话框编辑器 dialog editor 中再次建立你的项目. 在项目编辑器模式下输出 output window 在控件编辑器窗口下弹出.

现在你的项目包含了一个新的控件, 你作为程序员的工作就是实现所使用的槽的功能，并最后加入你可能需要的其他方法.

参见 KDevelop 编程手册, 对话框编辑器以获取关于控件属性及资源代码生成的更多信息.

# 12. 内置调试器

## 12.1 设置

KDevelop 的缺省设置是使用内置的调试器.你可以通过选择选项的"配置 KDevelop" 菜单来改变.

如果你要使用其它的调试器,点击"使用外部调试器"并输入其名字.参考你的调试器的文档看它是怎样运行的.

如果你选择内置调试器, 还有一些附加的选项:

- 显示静态成员: 显示静态成员使 gdb 在 kde 和 qt 中产生数据时变慢.它会改变 QString 及其友类所依赖的'签名'.但是如果你的确需要调试这些值就选上它.
- 显示 mangled 名字: 在显示反汇编代码时,你可以选择是看 mangled 名字,当然,非 mangled 的名字更易读.
- 在载入库中设置断点: 这将尝试在载入库中设置断点.如果 GDB 没有看到通过'dlopen'方式载如的库,那么就拒绝在该处设置断点.我们可以让 gdb 停在载入的库中并设置断点.参看 共享库与断点以获取更进一步的信息,如果你不是用"dlopen"方式载入库,关闭此选项
- 允许浮动的工具条: 作为对菜单和按钮的补充,它使你能够使用浮动的工具条来控制调试器. 该工具条在调试 GUI 应用程序的时候尤为有用.具体的细节参见 浮动工具条.
- 允许为程序的 I/O 使用分离的终端: 使你在应用程序包含从终端输入的代码(比如 cin,fgets 等)时能够输入.如果你在程序中使用了终端输入就勾上此选项.

## 12.2 使用内置调试器

### 12.2.1 树视图和输出视图窗口中的变化

如果你选择内置的调试器, 树视图和输出视图中会增加四个标签.

#### 12.2.1.1 树状视图窗口中

- VAR 标签: 暂停程序在当前位置的局部变量的树状视图.同时,你可以看见在调用该函数的函数中的变量.VAR 标签还包含了观察窗口.你可以查看全局变量或某一个局部变量而不是一大串局部变量的值.

#### 12.2.1.2 输出视图窗口中

- 断点 : 断点的列表及其状态.
- 帧栈 : 调用栈.
- 反汇编 : 当前被执行的机器代码.

### 12.2.2 在面板中和调试按钮的变化

#### 12.2.2.1 面板中

在你启动调试器后会有两个普通和可按下的按钮来控制调试功能.

#### 12.2.2.2 调试菜单中

党开始调试时, 10 条控制调试的菜单变为可用.

已实现的功能:

- 运行: 从当前位置继续程序执行.
- 运行到光标处: 执行程序直到光标的当前位置
- 单步跳过: 执行一行代码并停在同一源文件的下一行.这会运行任何遇到的函数直至其符合上述条件.
- 单步跳过指令: 象上面一样,执行一条机器指令.
- 单步进入: 刚好执行一行代码.即如果存在,你将"单步进入"函数.
- 单步进入指令: 象上面一样执行一条机器指令.
- 单步跳出: 运行到当前栈帧(函数)结束.
- 浏览器: 允许对数据进行各种查看. 当前实现的查看有:
    1. 内存地址

2. 反汇编代码
3. 当前寄存器
4. 当前库

- 暂停: 暂停程序执行.
- 停止 Stop: 停止程序执行并退出调试器.

## 12.2.3 细节

### 12.2.3.1 断点

在源文件中可以设置断点或观察点.两种类型的断点都可以在任何时间设置,而观察点只在变量的局部域中有意义.当你处理全局变量的时候,观察点更为有用.

### 12.2.3.2 设置/取消断点

只需要点击一下就可以设置/取消断点. 在编辑器中你需要设置断点行左边的 ″图标″边界上点击. 再次点击就取消了该断点.

### 12.2.3.3 通过菜单控制断点

在断点列表或编辑器的图表边界中的断点上点右键, 会出现一个关于断点的菜单. 你可以通过它来删除断点, 清除所有的断点或者编辑断点.

### 12.2.3.4 编辑断点

使用上面的菜单现实断点编辑对话框. 包括以下一些域:

1. 条件: 输入 gdb 中断程序执行的条件.
2. 忽略次数: 你希望 gdb 在停止程序执行前忽略该断点多少次.
3. 激活 Enable: 激活以后,gdb 将停于此处.如果禁止 gdb 则忽略.

### 12.2.3.5 清除所有断点

删除为该程序设置的所有断点

### 12.2.3.6 设置/取消观察点

在变量视图中的某个变量上点击鼠标右键.你可以通过现实的弹出式菜单在局部变量上设置观察点.该功能局限于局部变量所在的域.党变量超过了其有效域,观察点被删除.

警告:这可能会带来问题,所以在为局部变量设置观察点时要小心.

观察点也可以通过右击前一个遇到的观察变量并选择″设为观察点″.

### 12.2.3.7 设置/取消观察变量

在变量视图的底部有一个"观察"域,你可以在这里输入希望显示在列表中的变量名字. 输入变量名并按回车或点击旁边的"添加"按钮.鼠标右击树状试图中的观察变量会出现弹出式菜单,你可以通过它将观察变量删去.

你也可以在编辑窗口中的某个变量上点击鼠标右键,通过显示的菜单选择"查看:变量名" 来添加观察变量.

### 12.2.3.8 改变变量的值

通过观察变量实现.如果你有个变量叫"test",在观察域中输入"test=5"并将其添加到列表中.注意,"test"在程序每次被断点中断时都会被设为 5,所以你一旦设置了该变量, 往往需要从视图中删除掉.

## 12. 3 浮动工具条

浮动工具条是内置调试器的一个特征,它使你调试 GUI 应用程序变得舒适.该工具条要么在所有窗口的上面,要么在面板中.在面板中的时候,你可以点击按钮来执行程序.其执行的功能是"单步调过".

除开 已实现的功能, 浮动工具条提供两个附加的功能:

- 设置焦点于 kdevelop： 将焦点设置在 KDevelop 上
- 设置焦点于应用程序 ： 把焦点设置在当你按下"将焦点设置在 KDevelop 上"时拥有焦点的窗口上.这是一种折衷,应该把焦点设置在被调试的应用程序上,但这很难做到. 如果任何人有更好的方法实现这个功能,请告诉我们.

当 gdb 中断程序时,可能是碰到了断点,我们高亮显示"设置焦点于 KDevlop". 我们并不将焦点自动的切换到 KDevelop,这样你可以在该处看一下应用程序窗口. 点击"设置焦点" 按钮切换到 Kdevelop 或者按工具条上的按钮执行你选择的功能.

## 12.4 共享库和断点

共享库和断点有一个可以得到合理解决的问题.这个问题是:gdb 不会接受一个没有打开的共享库中的断点,虽然该共享库即将会用 dlopen 打开.

解决办法是让 gdb 告诉我们一个共享库什么时候被打开的.我们通过设置 "stop-on 1"来实现.这意味着,当用户设置一个断点的时候,我们把该断点标记为挂起,如果 gdb 告知中断成功则标记为活跃,否则保持其挂起状态.下一行代码将被"继续".

这就是"懒断点"

然而,当你使用"单步跳过"命令并且单步跳过的那行会载入库时可能会导致问题发生. 这会在载入库时引发中断.并且,一般调试器会"继续"(即运行到下一个断点或代码结束).可以用户希望程序停在下一行,所以在这个位置上,我们并不继续,而是让它留在那里(其中的某处会有一个 dlopen 命令).这不太和谐,但是没有更好的办法了.

# 13. CVS 集成

Kdevelop 还通过 CVS 存储器管理你的项目. CVS (concurrent version control) 的作用是让开发小组能够独立的使用同一个资源树并将改动合并到存储器. 通常, 存储器在服务器中定位. 最初的存储器包括项目所含文件的基础设置; 而目录和文件是由开发者加入 CVS 或从中去除的.

从局部来说, 开发者操作的是他的 CVS 树的拷贝. 他把他的改动编写到文件并测试, 如果他的项目仍然工作正常,至少在某种程度上不完整的资源没有阻止其他开发人员在收到改动后继续正常的操作 CVS 存储器.

## 13.1 创建仓库

当用 KDevelop 生成新的项目时,你可以直接将该项目导入为 CVS 模块.为此, 你必须安装 CVS (也许已经安装了,如果没有,看一下你的安装盘或者找你们的系统管理员).

在应用程序向导 KAppWizard 的第三页(VCS 支持),选择"CVS".随后该页的编辑框就变亮, 你可以指定导入的参数,下面将解释给出了选项.应用程序向导将把项目的源文件树做为一个 CVS 模块插入到 CVS 的根目录下,同时在你本地生成 CVS 的一个拷贝作为新的项目.

**注意**:为生成起始仓库,你必须在 CVS 的根目录拥有写权限!而且该 CVS 根目录必须和你运行的 KDevelop 在同一台机器上. 如果你要在一台公用的 CVS 服务器上使用仓库,那要么联系该服务器的管理员在服务器上用 KDevelop 建立项目,要么就按照以下步骤做:

- 在本地创建项目
- 将项目作为一个 CVS 模块导入到 CVS 服务器中
- 删除本地项目的源文件树
- 从 CVS 上获取项目源文件
- 载入你从 CVS 获取的项目后,在下述的选项对话框中打开 CVS 支持.

要获取更进一步关于 CVS 的信息,请参考你本地系统中关于 cvs 的帮助,即 man cvs

下面我们讨论用应用程序向导生成仓库时碰到的若干选项

**VCS 位置**:

这是项目源文件树将被导入,提交,获取,更新时的 CVS 根目录.

VCS 仓库

这是你的项目在仓库中的模块名.由于它和项目的目录名相同,你可以在此改变.

标签

这是整个分支的标签.

日志信息

是新模块的第一条日志信.当向仓库添加或提交时,你将被询问一条日志信息以以记录你对仓库中源文件的更该.

Release 标签

指定你的项目源文件树放在那一个标签下面,即你可以在 CVS 根目录下用相同的模块名生成多个项目,彼此间用这个标签区分.所以你可以同时做相同项目的不同分支.例子:KDE 2 的标签是(缺省的) HEAD, KDE1.1.2 的标签是 KDE_1_1_1_RELEASE.

在用应用程序向导创建完你的项目以后,CVS 支持选项已经设置,所以你可以象章节" 使用 CVS 命令"中所描述的,提交或者添加更改.

## 13.2 激活 CVS 支持

由于 CVS 支持大多只是专家或公司及在 KDE CVS 服务器下维护其项目的 KDE 开发者所需要, 是否使用 CVS 开发命令可随意. 要使 CVS 命令可用, 请先打开项目-选项对话并将选择的版本控制改变为"CVS"。

## 13.3 使用 CVS 命令

以下命令通过上下文菜单在 LFV 及 RFV 中可用:

对于未包含于存储器的文件:

- 加入存储器 Add to Repository: 将文件加入存储器. 文件将被设置为加入并将由包含了所加入文件定位的目录的 commit 命令传入存储器.

对于存储器中已包含的文件:

- 更新 Update: 随 CVS 版本更新选中文件
- 提交 Commit: 提交所选文件到 CVS 存储器
- 从存储器中去除 Remove from Repository: 从存储器中去除文件

对文件夹:

- 加入 Add: 将文件夹加入存储器
- 去除 Remove: 从存储器中去除文件夹
- 更新 Update: 递归的更新文件夹以与存储器同步
- 提交 Commit: 递归的将任何改动提交到存储器

CVS 命令都需要与 CVS 服务器进行网络连接.当激活某一命令, 你将看见一个对话,同时命令被提交至 CVS 服务器,恢复输出. 因此你可以控制实际发生的情况及命令是否成功.

由于 KDevelop 的 CVS 命令只与 cvs 系统命令及其命令行选项协作,对于使用 cvs 特色你应该毫无问题.

# 14. 一般设置

本章将描述你将如何设置你的 KDevelo 工作的个人参数. 以下叙述的所有设置可以在选项菜单的相关条目中找到.

## 14.1 设置工具"Tools"菜单

由于 Kdevelop 在其用户界面中支持第三方次程序的使用, 你可以设置 configure 符合你应用开发所需要的任何程序. 这可通过在工具菜单"Tools"-menu 中将程序加入已预先定义的程序来完成. 要改变工具菜单, 请从"选项"菜单中选择工具"Tools". 这一对话还允许指定条目名称, 程序及?想在执行中传递的另外的命令行选项. 要从目录中除去一个程序, 请选择条目名称和"删除". 要加入一个程序, 先指定菜单入口, 这里 & 被用作菜单-加速器; 你可以对比已设置的入口和入口列表. 选择二进制并传递你的命令行选项. 然后点击"加入", 此入口就被加入列表. 在离开设置对话 configuration dialog 以后, 工具菜单会自我更新, 这样无须重新启动 Kdevelop 新的设置就可以使用了.

## 14.2 文件浏览器选项

逻辑化的文件浏览器可以通过上下文菜单完整设置。由于此举的目的是逻辑的分开文件以更好的浏览整个项目, 最常用的设置之一是创建文件组。它们可以通过在树的根部显示的项目按钮上右击鼠标来打开上下文菜单进行设置。该菜单提供：

- 新文件 New File: 打开新文件 New File 对话。等同于菜单条命令 "文件"-"新建"
- 新类 New Class: 打开类生成器以创建一个新的类。等同于菜单条命令"项目"- "New Class"
- 新组: 打开对话建立一个新组。可在此设置组名及该组所显示的项目文件 project files 的过滤器。
- 显示相对路径: 显示文件及其路径名，路径从主项目目录开始（如果检测到）；否则只显示文件名。

在主文件夹，相关的上下文菜单提供：

- 新组: 打开新组对话，如上下文菜单所述。
- 取消组: 从 LFV 中取消组。
- 属性: 打开组的属性。可在此通过由 commas 分开的 wildcards 列表编辑文件过滤器。

## 14.3 Kdevelop 设置

**Make-command:** 一般选项对话允许你进行 Kdevelop 的一般设置。首先，你应该在你的系统上将 make-command 设置为可用。如果选中的程序不存在,下一次 KDevelop 会警告你你正在激活 make 命令。 **自动保存 Autosave:**

如果选中自动保存, Kdevelop 会定期对所有改动过的文件进行保存。自动保存的时间范围可设置为 3, 5, 15 或 30 分钟。

**自动切换 Autoswitch:**

如果选中自动切换, Kdevelop 窗口会根据上下文的使用切换为开和关, 例如，如果你切换到帮助菜单中的一个文档, 文档浏览器将打开, 文档树和输出 output 窗口则关闭。启动:

要启动 KDevelop, 你可以在 Kdevelop 装载期间激活/不激活 start-logo 的显示。而如果你不喜欢最近的项目在启动时被打开, 你可以取消缺省。

## 14.4 改变键盘快捷方式 Keyboard Shortcuts

键盘键设置对话允许你设置 Kdevelop 的键盘键绑定。注意，通用键均可在 KDE 控制中心进行设置, 例如打开文件和打印。 某键的功能可以通过选择菜单条目进行设置。然后设置可以通过校验数值，例如 Alt / Ctrl 键等来改变。

## 14.5 文档

### 14.5.1 目录

要想通过设置使文档浏览器 documentation browser 正常工作, KDevelop 需要一些关于 HTML-documentation 在系统上位置的信息。因此, 文档路径属性对

话需要 HTML 中 Qt-在线文档的路径及到 KDE-库文档的路径。

通常, Qt-文档 Qt-Documentation 位于 Qt 安装的相同目录中；例如，如果 Qt 位于 /usr/local/qt, 你必须输入的路径即为/usr/local/qt/html。对于 KDE-文档, 你必须将目录设置为文档的根部, 确保所有的 KDE-库文档位于相同的目录中。两个路径都可以通过按下相关按钮, 显示路径设置来选择。如果你的系统部包含 KDE-库文档, 你应该限输入下一个设置对话 configuration dialog, 升级 KDE 文档 KDE-Documentation。这会按你选择的路径生成文档,并自动设置 KDE 库文档路径。

## 14.5.2 选项

### 14.5.2.2 升级 KDE-Documentation

对于那些没有近期 KDE 库 KDE-libraries 的用户, 尤其是安装在系统上的文件的文档, 升级的 KDE 文档 KDE-Documentation 对话会生成一个新的或升级已有的文档。这一功能要求你的系统已经安装了 KDoc 及 qt2KDoc, 包含于 KDE-SDK 包内。首先, 你必须设置你近期 kde 库资源的路径, 这不是包含 KDE 的路径! 只需输入资源的路径, 比如:/home/rnolden/kdelibs-1.1/.

接着, 你可以选择三种不同的安装模式, 如下:

- 删除旧文档并安装于近期的文档路径: 这假定你已经安装了一个文档且位于输入文档路径对话的路径下。这将删除所有文档并安装新生成的文档到近期路径。
- 删除旧文档并安装于新的文档路径: 这会导致象上一种模式一样删除旧文档, 但你可以选择建立一个新的文档地点。
- 不改变旧文档并安装于新的文档路径:推荐那些以前没有该文档或希望为老版本的 kde 库保留上一个文档的用户生成一个新的 kde 库文档时使用。

"新的 KDE 库文档路径"将为第二和第三种安装模式设置。推荐从头开始生成新文档的用户使用。在按下 OK 按钮后, Kdevelop 会在包含 KDoc 参考文件的文档路径中生成一个子目录 "KDoc-reference" 。首先, qt 库文档 classes 将被索引以由将为 kde 库生成的文档来连接 Qt 文档。因此你应该先设置好 Qt 文档路径以确保它可以由 qt2kdoc 找到，这一点很重要。最后, KDE 库将被索引，文档将前后对照着建立，以使浏览具有更强大的功能。

#### 14.5.2.1 创建搜寻数据库

创建查找数据库对话，可通过建立按钮进入，它允许编程人员创建数据库以交互的方式查找关键字。要创建和使用文档查找功能，你必须先安装 glimpse 4.0 程序。事先调整索引给定的 KDE 库文档及 Qt-Documentation 的选项，假定文档文件的路径已在选项菜单的 "文档路径"对话中设置。另外，索引可以包括用户可以通过" 另外的索引目录" 域自行设置的目录。在设置好另外的目录的路径后，必须按下"增加" 按钮以设置路径。一个以前设置的路径可以通过在路径域选择路径并按下"去除"从索引中去除。此外，用户可以选择三种不同的索引尺寸：极小的，小型的以及中型的。索引尺寸越大，索引文件越长。 但是，更大的查寻数据库中的查寻会更加快捷，因此我们推荐使用"中等" 尺寸。要使用查找功能，请参见 使用文档浏览器.

## 14.6 调试器

KDevelop 的缺省设置是使用 内置调试器 如果你希望使用其它的调试器,输入其名字.查看该调试器的文档,确定如何运行.

如果你选择内置调试器, 还有一些附加的选项：

- 显示静态成员：显示静态成员使 gdb 在 kde 和 qt 中产生数据时变慢.它会改变 QString 及其友类所依赖的'签名'.但是如果你的确需要调试这些值就选上它.
- 显示 mangled 名字：在显示反汇编代码时,你可以选择是看 mangled 名字,当然,非 mangled 的名字更易读.
- 在载入库中设置断点：这将尝试在载入库中设置断点.如果 GDB 没有看到通过'dlopen'方式载如的库,那么就拒绝在该处设置断点.我们可以让 gdb 停在载入的库中并设置断点.参看 共享库与断点以获取更进一步的信息,如果你不是用"dlopen"方式载入库,关闭此选项
- 允许浮动的工具条：作为对菜单和按钮的补充,它使你能够使用浮动的工具条来控制调试器. 该工具条在调试 GUI 应用程序的时候尤为有用.具体的细节参见 浮动工具条.

- 允许为程序的 I/O 使用分离的终端: 使你在应用程序包含从终端输入的代码(比如 cin,fgets 等)时能够输入.如果你在程序中使用了终端输入就勾上此选项.

## 14.7 设置路径

在路径栏输入 Qt 2.x 和 KDE 2 的安装路径以及在那里生成新项目.关于安装 Qt 2.x/KDE 2 以及将 KDevelop 配置为 Qt 2.x 和 KDE 2,请阅读 KDE 应用程序教程.

KDevelop 生成新项目的缺省位置是用户的起始目录.如果你需要改变,在缺省项目路径中输入你希望的目录.

# 15. 问与答

本节将叙述 Kdevelop 小组或其支持者在 Kdevelop 邮件列表中对用户在 Kdevelop 的当前版本的使用中遇到的问题的回答及一般臭虫报告。

## 15.1 臭虫报告 Bug Reporting

Kdevelop 的另一个改进是集成的通过 email 完成的臭虫-报告系统。如果你遇到一个臭虫，你可以通过你的 email-客户或臭虫-报告对话给 KDevelop 开发小组发送一个臭虫报告。所有的 臭虫报告将收集在 Kdevelop 互连网站点并可在 http://fara3.cs.uni- potsdam.de/~smeier/kdevelop/bugarchive/maillist.html 观看。你还可以通过发送一个空的以"预定 *your_email_address*"作为正文内容的 email 邮件至 kdevelop- bug-report-request@fara3.cs.uni-potsdam.de 来预订臭虫-报告有列表，以收到所有臭虫-报告。

要发送你的臭虫-报告，请通过你的邮件程序使用这个 email 地址。如果你想要使用 Kdevelop 进行直接的臭虫-报告，请从帮助菜单 Help-menu 中选择"臭虫-报告"。报告对话将显示，允许你输入所有关于你所发现的臭虫的必要的信息。在按下"确定"后，对话内容将自动发送到邮件列表中。

## 15.2 从哪里可以得到信息

**问**：如果我遇到了在 FAQ 文件及 KDevelop 手册中未曾包含的问题，我应该在哪里寻求帮助?

**答**：在任何一种情况下都可以在 kdevelop@fara3.cs.uni-potsdam.de 预订 Kdevelop 邮件列表来询问所有关于 KDevelop 的问题。发送一封头部为空，以"预订"为内容的邮件；然后你就可以参与讨论。所有问题都可以去那里叙述和讨论。如果你坚持那样，你可以得到开发者及所有遇到同样问题的用户的最大帮助，FAQ 也可以不短更新。

KDevelop 主页 KDevelop Homepage 位于 http://www.kdevelop.org ，它也包含了一个邮件列表 mailing-list 档案，允许你浏览以交送的邮件，因此你应该先去哪里看看，大多数问题都以由开发小组及用户所提及。

## 15.3 库和系统问题

**问**：错误的 JPEG 库版本：库为 61 版，需要 62 版

**答**：有两种方法。

1. 当 kde 库安装好后，它会为 jpeg 库安装头文件，这些是 61 版本，然而大多数发布的的软件（红帽子）使用的是 62 版本的库。要确定这一点只需从/opt/kde/include 中移动 jpeglib.h。应该拾起包含了 62 版本需要的文件的 pukka 。但是看看它上面的错误信息也许是另一种方法，无论如何你要确保你只拥有头文件及库的一种版本，而且它们是一致的。使用定位命令来证实我拥有正确版本的库和头文件十分有用，例如：updatedb 定位 libjpeg 定位 jpeglib
2. 你必须重新编译 kde 支持(不用 jpeg 压缩格式)库 (./configure --使用-libjpeg -使用-libgif).

**问**：

```
make[2]: Entering directory `/usr/local/src/kdevelop-0.3/po'
cd .. && automake --gnu --include-deps po/Makefile
aclocal.m4: 2709: `AM_PROG_INSTALL' is obsolete; use `AC_PROG_INSTALL'
make[2]: *** [Makefile.in] Error 1
```

**答**:对于 automake-1.4/automake-2.13 用户：只需手动运行 "aclocal"就可以编译它。

**问**：如果 configure 说我需要 giflib23 我应该怎么办？

**答**：试着使用一个更新的 kde 支持的快照，或者安装另外一个 giflib。

**问**：我如何转换 KDevelop 0.2 项目为 0.3 项目？

**答**：请在 configure.in 中把 AC_OUTPUT 改变为在线版本。

例如：旧版本：

```
AC_OUTPUT(Makefile \
kdevelop/kwrite/Makefile \
```

kdevelop/templates/Makefile
)
新版本:
AC_OUTPUT(Makefile kdevelop/kwrite/Makefile
kdevelop/templates/Makefile)

**问:** 在由 KDE 1.1 使用 SuSE Linux 时我得到了以下连接器错误 Linker errors, 怎样才能使 Kdevelop 正确连接?

/usr/lib/libqt.so:
warning: multiple common of `QArrayT<char> type_info node'
ckdevelop.o: warning: previous common is here
ckdevelop.o: In function `CKDevelop::slotFileSaveAll(void)':
ckdevelop.o(.text+0x784): undefined reference to `kdebug(unsigned
short,
unsigned short, char const *,...)'
ckdevelop.o(.text+0x839): undefined reference to `kdebug(unsigned
short,
unsigned short, char const *,...)'
ckdevelop.o(.text+0x89d): undefined reference to `kdebug(unsigned
short,
unsigned short, char const *,...)'
ckdevelop.o: In function `CKDevelop::slotFileSaveAs(void)':
ckdevelop.o(.text+0xd28): undefined reference to `kdebug(unsigned
short,
unsigned short, char const *,...)'
ckdevelop.o: In function `CKDevelop::slotFileClose(void)':
ckdevelop.o(.text+0x1216): undefined reference to `kdebug(unsigned
short,
unsigned short, char const *,...)'
ckdevelop.o(.text+0x1263): more undefined references to
`kdebug(unsigned
short, unsigned short, char const *,...)' follow collect2: ld
returned 1
exit status make[2]: ***

[kdevelop] Error 1  make[2]: Leaving directory
'/home/LinuXDaten/Programme_Updates_Packete/KDE_Updates/Kdevelop_actual_snapshot/kdev
elop-0.3/kdevelop'
make[1]: *** [all-recursive] Error 1  make[1]: Leaving directory
`/home/LinuXDaten/Programme_Updates_Packete/KDE_Updates/Kdevelop_actual_snapshot/kdev
elop-0.3'
make: *** [all-recursive-am] Error 2

**答**：如果你由使用 SuSE Linux，你必须重新编译 kde 库且不使用 SuSE 的补丁，并重新安装，或从 ftp://ftp.suse.com 获得 kde 库的升级版本

## 15.4 使用中的问题

**问**：我发现 Kdevelop 并不允许使用 delete 键(或退格键来删除标注的文本)。

**答**：在"选项"->"编辑器"中确认"输入时删除" 被选中，然后就可以进行退格和删除操作。

**问**：如果我在项目中加入了文件，加入的文件会被自动包含并编译吗？

**答**：是的，它们包含于 Makefile.am 内，如果你使用了"全部重建"(./configure 升级 Makefiles)，你新加入的文件也会包括在内。

**问**：如果我移动了一个文件，我会得到一些奇怪的连接器 Linker 的信息。我的项目究竟这么了？

**答**：如果移动的文件是一个头文件，那将由自动进行 automoc（自动在头文件上运行 Qt-Meta-对象--编译器)，你移动过的头部仍然以一个 moc-生成的 *.moc.cpp 文件表示并编译。移动相关的*.moc.cpp 文件并重建项目。