DoSmall 帮助文档

关于 DoSmall

DoSmall 是一个使用 Qt 开发的针对 Linux 操作系统的 C/C++程序开发 IDE，支持 GPL v3.0 协议.

为什么叫 DoSmall:

有句话叫"Think big，do small."，意思是目标要远大，而做事情的时候则要把小事情做好。是的，开发软件的时候我们的目标是远大的， 而在软件开发的过程中则要把每一个细节都做好，因为软件的每一个部分都是软件质量的基础，而 DoSmall 面对的就是软件开发过程， 所以取名为"DoSmall"。

DoSmall 目前含有以下几部分：

1. 程序编辑器：支持代码编辑、保存，显示行号等基本编辑器功能。
2. 项目管理：可以创建 C 或者 C++类型的项目，可以用项目浏览器管理项目和项目目录下的文件。
3. 程序编译和执行：根据项目配置编译和运行程序。
4. 终端控制台：集成了终端控制台， 用户可以在上面 shell 程序。

使用 DoSmall 需要的系统软件：

1. gcc 或者类 gcc 的 C 语言编译器。
2. g++ 或者类 g++ 的 C++语言编译器。(如果要开发 C++程序的话)
3. make
4. xterm

DoSmall 的目标:

用一句话概括—— 将 Linux 下重要的开发工具集成到 DoSmall 中，做一个让用户快乐的在 Linux 下开发程序的 IDE。

具体可以分为以下几点：

1、强大的代码编辑器，可以选择 Emacs 或 Vim 的代码编辑方式。

2、强大的编译器,充分发挥 gcc、g++的强大功能，将它们的设置和使用可视化。

3、强大的调试器，将 GDB 调试器可视化。

4、强大的项目管理器，发挥 make 做项目管理的优势，灵活的项目管理。

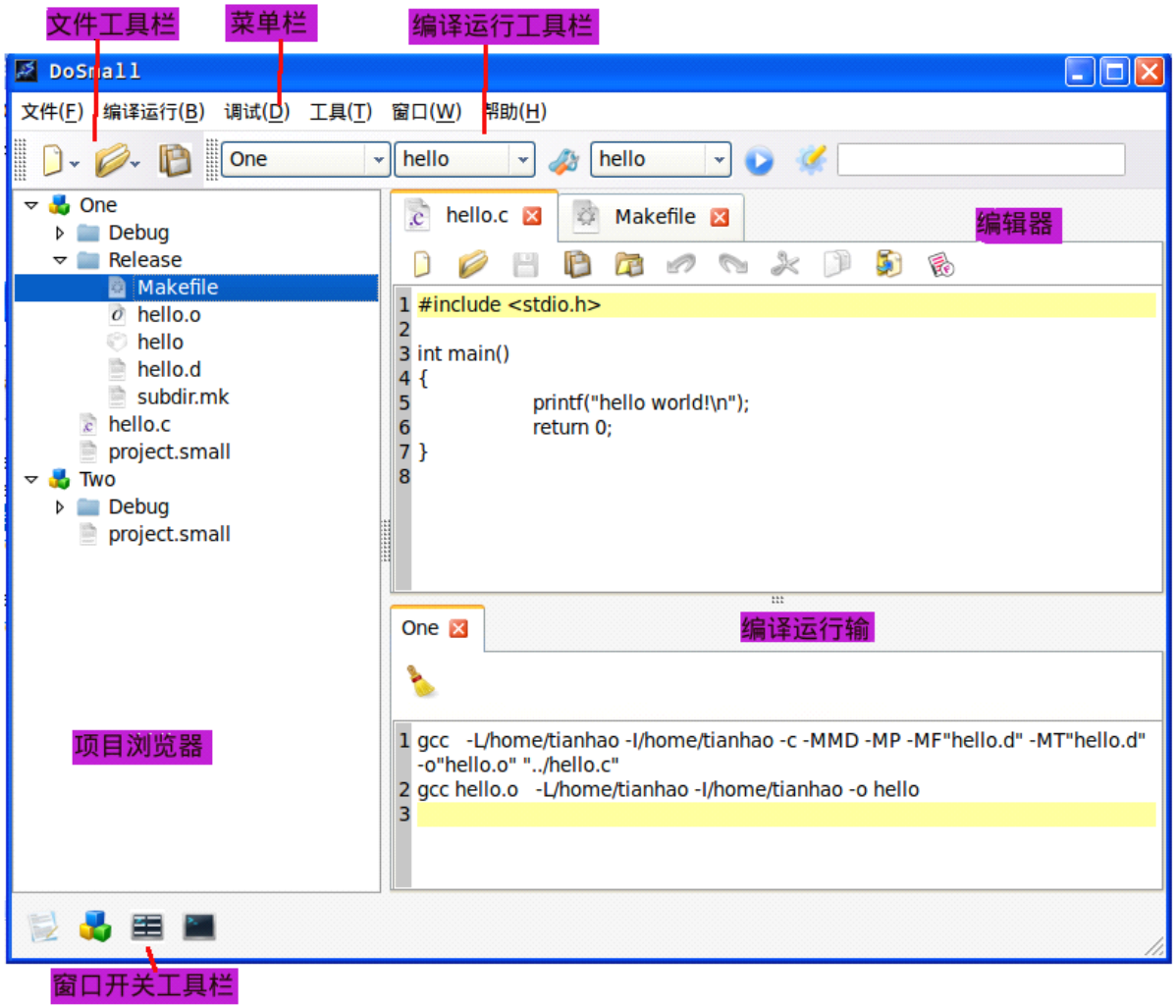
5、强大的版本控制器，使用 Git, Subversion 等版本控制器，可以在本地和网络上做版本

控制工作。
6、软件打包可视化，可以在 IDE 中将己开发的软件打包成 deb、rpm 等格式的软件包。
7、联机协作开发。

目前参与开发的人员：

天浩　tianhaolsk@gmail.com

软件整体视图：



项目浏览器

点击菜单栏的【窗口】->【项目浏览器】，即可打开或关闭项目浏览器。项目浏览器可以浏览和管理项目的文件。

右击项目浏览器里的节点会显示操作菜单，在菜单含有以下操作选项：

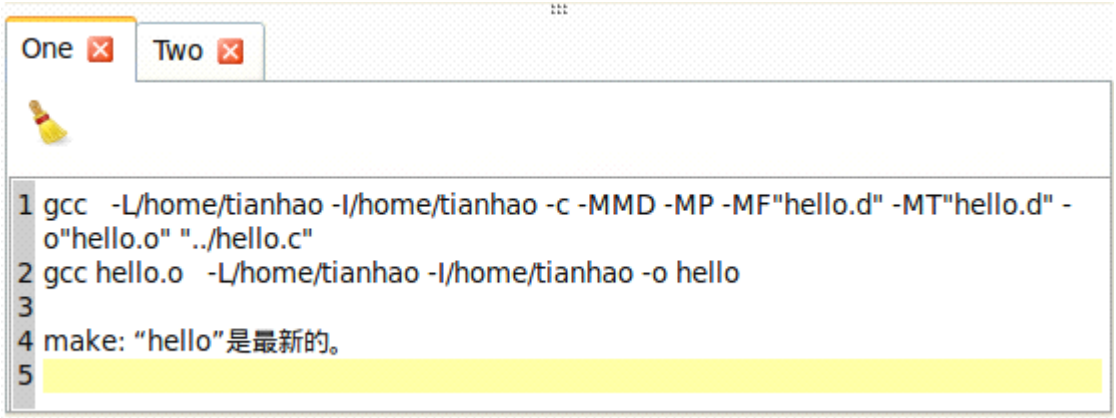
1. 创建文文本件/源代码/文件夹。
2. 打开文本文件/源代码文件。
3. 删除文件/项目。
4. 关闭项目/打开已关闭项目。
5. 编译项目/运行项目编译得到的程序。

编译运行输出

点击菜单栏的【窗口】->【编译输出】，即可打开或关闭项编译输出窗口。编译输出窗口与终端控制台在同一个窗口区域， 打开编译输出窗口，终端控制台窗口将隐藏，反之则编译输出窗口隐藏。这样可以节省不必要的空间浪费。

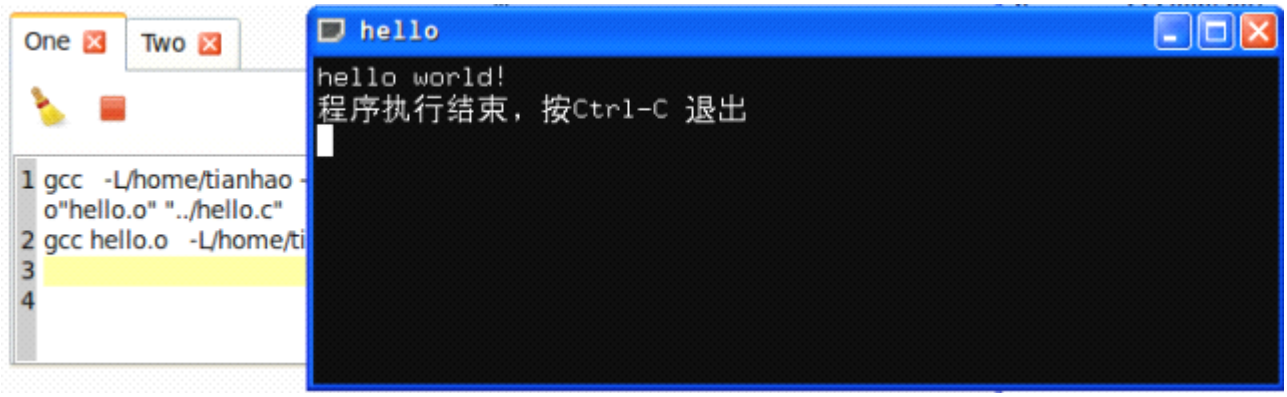
当编译程序或者运行程序时，会产生一个输出窗口，一个项目对应一个输出窗口，在输出窗口中可以查看项目的编译过程，以及编译错误。

如下图：



输出窗口有一个工具栏，当运行一个程序时，在该工具栏中会添加一个按钮与该程序执行的进程的按钮，点击该按钮可以退出与之对应的进程，当进程退出时，与该进程对应的按钮也在工具栏中消失。

如下图：



上图中，红色按钮与当前运行hello 进程对应， 点击红色按钮即可退出hello 程序，当hello 进程退出时，该红色按钮也消失。

终端控制台

点击菜单栏的【窗口】->【终端控制台】，即可打开或关闭终端控制台。编译输出窗口与终端控制台在同一个窗口区域， 打开编译输出窗口，终端控制台窗口将隐藏，反之则编译输出窗口隐藏。这样可以节省不必要的空间浪费。

如下图：

如果上次运行的终端控制台退出了，在打开终端控制台时，会新建一个控制台窗口。用户可以在终端控制台执行 shell 程序。

创建 C/C++项目

1. 【文件】->【新建项目】，如下图：
2. 在【项目类型】选择栏中选择项目类型，C 类型的项目，编译时用到c 语言编译器，C++类型的编译时用到c++语言编译器。
3. 在【项目名称】编辑区中输入项目名称。
4. 在【项目位置】中输入项目文件夹创建的位置。
5. 在【项目文件夹】编辑区输入项目文件夹名称， 在输入项目名称的时候， 项目文件夹编辑区根据项目名称显示“项目位置/项目名称 “。
6. 点击【确定】，即在项目位置下创建了【项目文件夹】文件夹，并创建了项目配置文
7. 件” project.small” 。在项目浏览器中添加了该项目。

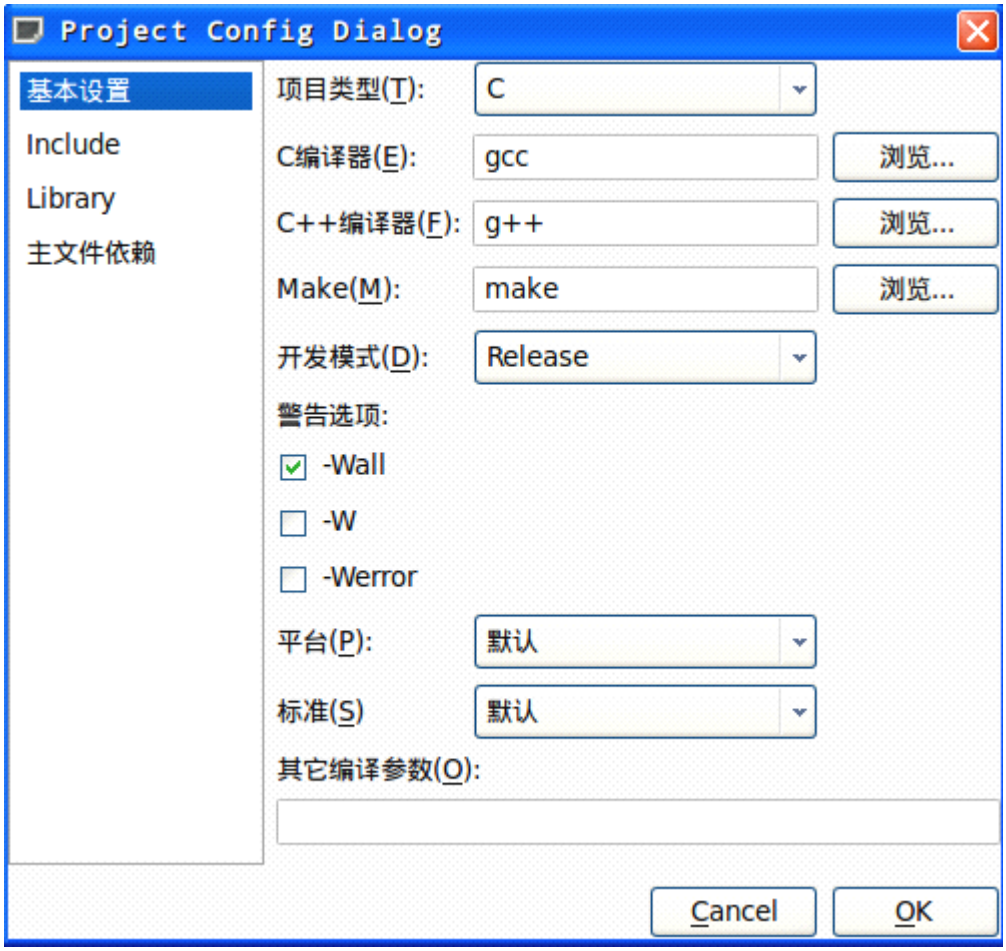
项目配置

配置方法：

1. 在项目浏览器中右击项目的根节点选择【配置项目】。
2. 在“build 工具栏”中点击【配置项目】按钮，会弹出一个项目配置窗口。

基本配置

如下图：



项目类型分为**C**和**C++**， 他们的区别是，C 类型的在编译时统一发使用C 编译器，C++类型的在编译时.c 后缀名文件用C 编译器，.cc、.cpp、.cxx 等后缀名文件用C++编译器，最后将.o 编译成可执行文件时用C++编译器，所以如果系统没有C++编译器， 则不要选择 C++类型。

C编译器在默认情况下是操作系统里默认的gcc。

如果操作系统里有其它的gcc 或类gcc 编译器(类如前Sun 公司的修改gcc 后的cc)，可以选择其它编译器。**需要主意的是**， 如果该编译器不在操作系统默认PATH变量的路径中，则必须是绝对路径(如: /home/user/bin/gcc)， 否则编译时会找不到该编译器。

C++编译器，在默认情况下是操作系统里默认g++。

同 C 编译器一样，可以选择其它 g++或类 g++编译器。

Make在默认情况下是系统里默认的make，也可以选择系统里的其它make。

开发模式分为Debug 模式和Release 模式。

默认情况下是Debug 模式。它们的区别是：Debug 模式下编译是添加了“-g”选项，也就是可执行文件中包含了调试信息， Release 模式下则没有该选项，Debug 模式下Build的.o 文件以及其它如Makefile 文件保存在项目根目录下的“Debug”文件夹中， 而Release 模式则保存在项目根目录下的“Release”文件夹中。当要编译时添加调试信息时选择Debug 模式，当程序开发完成不需要调试时选择Release 模式。

警告选项有“-Wall”、“-W”、“-Werror”。

默认情况下是“-Wall”，“-Wall”选项打开了针对许多常见错误的警告，所以最好选上它，“-W”选项是类似“-Wall”的通用选项，它针对a selection of 常见编程错误产生警告。“-Werror”选项将警告转化为错误。

平台目前是只有默认。

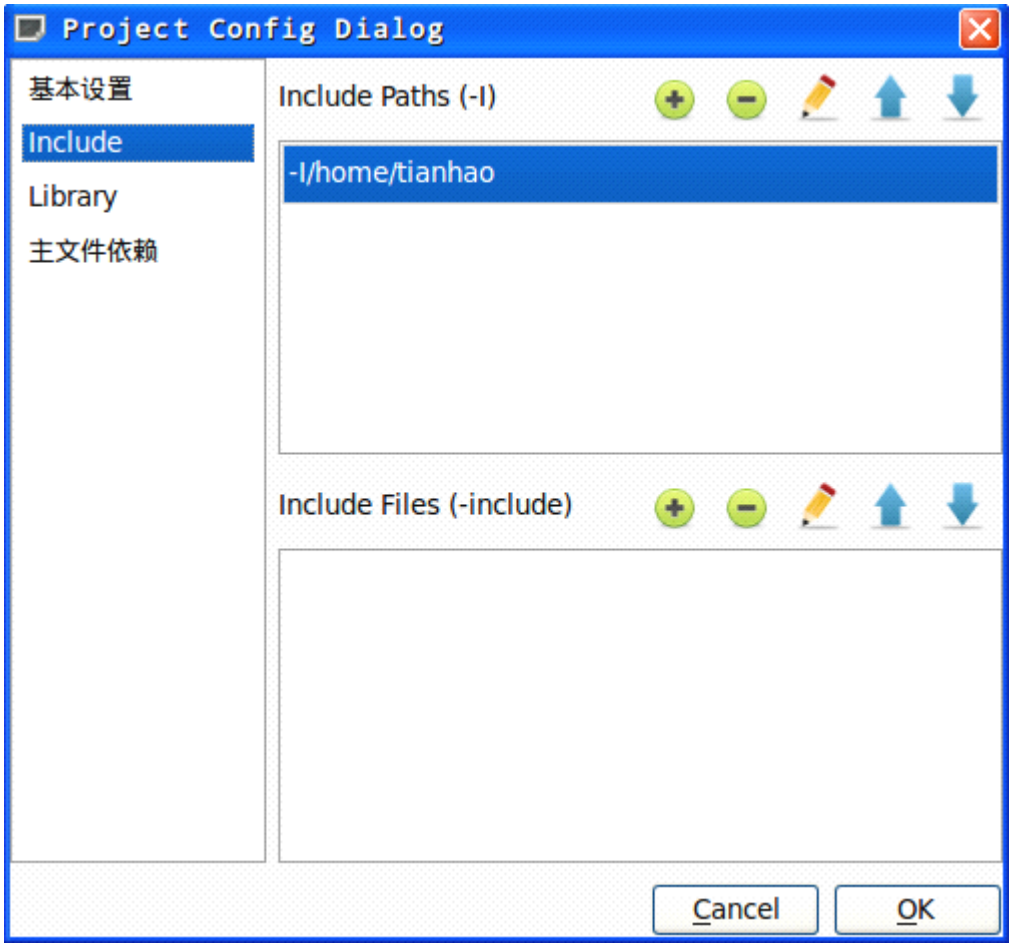
标准有“-ansi”、“-ansi -pedantic”、“-std=iso9899:1990”、“-std=iso9899:199409”、“-std=9899:1999”、“-std=gnu89”、“-std=gnu99”等选项。

默认情况下是gcc 的默认标准， 当要使程序符合某个标准时，可以选择相应的标准选项。

其它编译选项是由用户自己输入的选项，编译时会将它加入到编译选中，例如开发Gtk程序时，可以在其它编译选项中填入”`pkg-config --cflags --libs gtk+-2.0`“。

Include 配置(头文件配置)

如下图：



gcc 和g++编译程序时，使用“-I头文件路径”添加头文件搜索路径， 使用“-include”添加头文件。点击Include Paths(-I)右侧的【添加】【删除】【编辑】【上移】【下移】可以设置项目的头文件搜索文件路径。点击Include Files(-include)右侧的【添加】【删除】【编辑】【上移】【下移】可以设置项目的头文件。

点击Include Paths(-I)右侧的【添加】时，会弹出一个窗口让用户输入头文件目录路径，如下图：

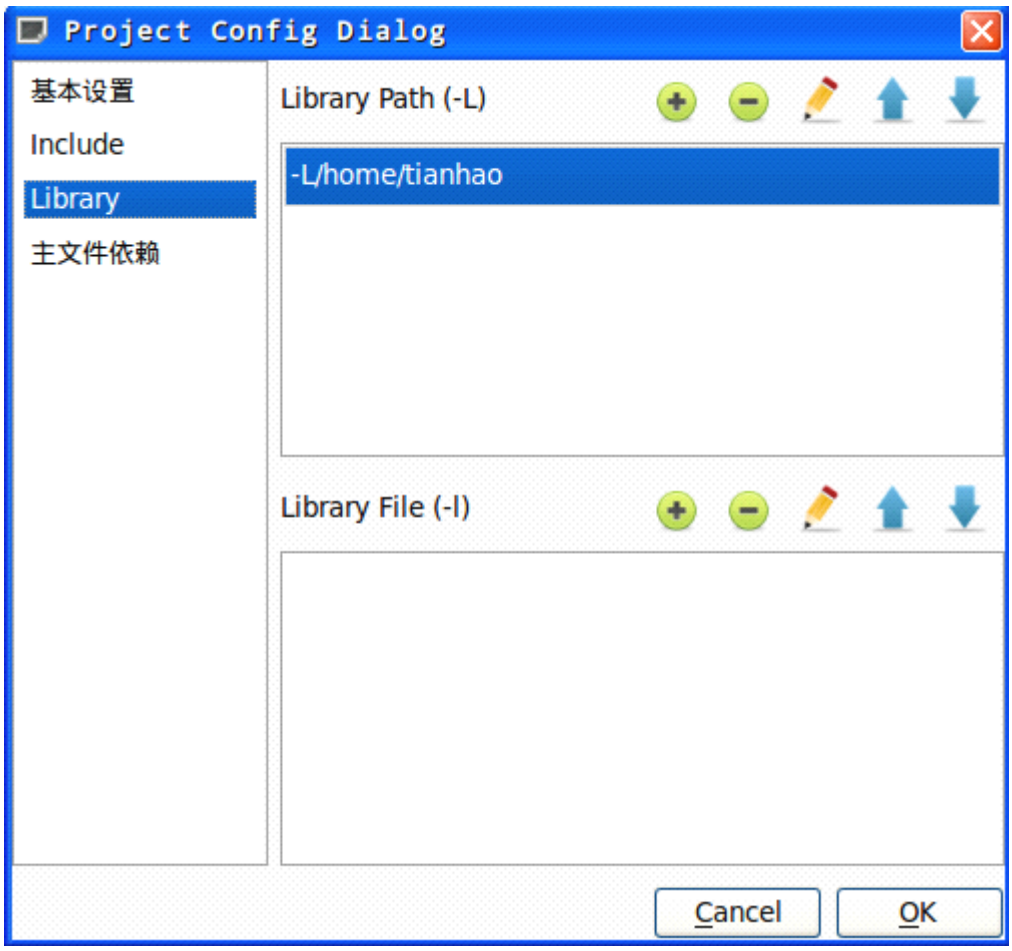
在文本框中可以手动填入多个头文件目录路径，两个路径之间以” ：“隔开，可以在路径前面加上"-I"，如果没有添加，程序会自动添加它。点击【浏览(F)...】会弹出一个窗口让用户选择目录，用户选择的目录将添加到文本框的末尾。点击【确定(O)】将文本框中的头文件路径添加到头文件目录列表中。

点击Include Paths(-include)右侧的【添加】时，会弹出一个窗口让用户输入头文件路径，如下图：

”添加文件“和”添加路径“的操作一样。不同的是”添加文件“添加的是文件路径。点击【确定】会将文本框中的文件路径添加到头文件列表中。

Library 配置(链接库配置)

如下图：



项目连接库路径是使用“-L 库文件路径”将该路径添加到编译过程中的。点击Library Paths(-L)右侧的【添加】【删除】【编辑】【上移】【下移】可以设置项目的库文件路径。

项目库文件是使用“-l 库文件”将连接库文件添加到编译过程中的。点击Library Files(-l)右侧的【添加】【删除】【编辑】【上移】【下移】可以设置项目的库文件。

点击Library Paths(-L)右侧的【添加】时，会弹出一个窗口让用户输入链接库的目录路径，如下图：

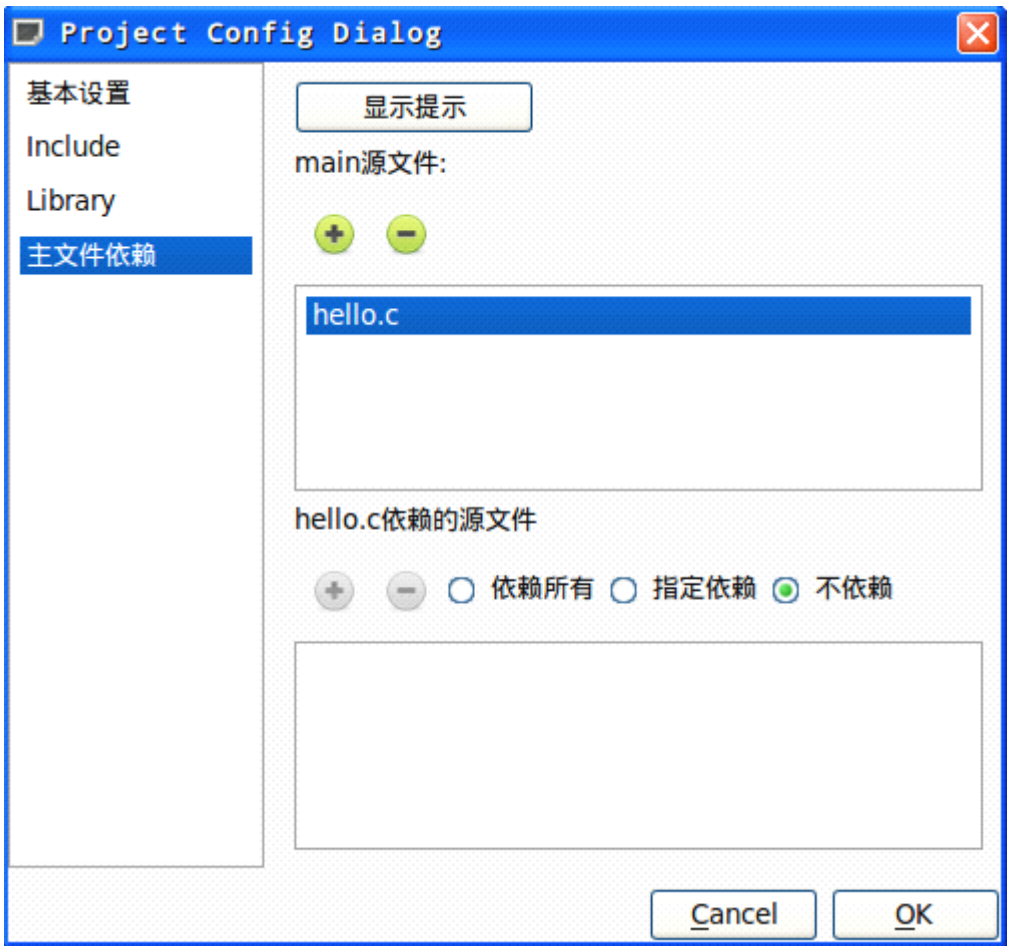
在文本框中可以手动填入多个链接库的目录路径，两个路径之间以”：“隔开，可以在路径前面加上"-L"，如果没有添加，程序会自动添加它。点击【浏览(F)...】会弹出一个窗口让用户选择目录，用户选择的目录将添加到文本框的末尾。点击【确定(O)】将文本框中的链接库的目录路径添加到链接库目录列表中。

点击Library Paths(-l)右侧的【添加】时，会弹出一个窗口让用户输入头文件路径，如下图：

”添加文件“和”添加路径“的操作一样。不同的是”添加文件“添加的是文件路径。点击【确定】会将文本框中的文件路径添加到库文件列表中。

主文件依赖

如下图：



先了解一下本文所用名称定义：

“main 源文件”——指含有main 函数的可以编译成可执行文件的源文件。

“源文件”——以“.c/.cpp/.cc/.cxx/.c++”为后缀的代码文件， 不包括头文件。

项目中允许有多个含main()函数的源文件对开发和测试都比较方便，为了缩小编译的范围，可以设置main 源文件依赖的其它源文件，注意以下几项:

1. 如果只有一个含main()函数的源文件，可以不设置依赖关系. 默认将所有源文件编译到一个名为项目名称的可执行文件。
2. 项目中如果有多个main 源文件，必须将main 文件添加到main 文件列表中，否则编译时会产生将多个main 文件编译到一个可执行文件中的错误。
3. 非main 源文件不要添加到main 文件列表中, main 文件不要添加到依赖文件列表中。
4. 不要添加头文件，头文件也不需要处理，因为编译时会自动处理源文件对头文件的依赖关系。
5. “依赖所有”表示该main 的源文件依赖所有非main 源文件，“指定依赖”表示依赖指定的源文件，“不依赖”则不依赖其它任何源文件。
6. 如果设置了对一个main 文件设置了“指定依赖”，则必须将该main 程序依赖的所有源程序添加到依赖列表中，如果依赖文件不全，则会出现找不到变量/函数/类等情况。
7. 所有源文件都必须放在项目的根目录或其子目录下，否则会无法编译该文件。

创建 Gtk 项目

创建项目的过程请见创建C/C++项目， 创建之后，在”编译运行工具栏“中选择创建的项目为当前项目，然后点击工具栏中的”项目属性“会打开项目配置窗口（参见“项目配置”），有两种配置方法：

1. 在”基本配置“页中的”其它编译参数“文本框中填入” `pkg-config --cflags --libs gtk+-2.0`“，点击确定即可。
2. 在 dosmall 的终端中输入并执行“ `pkg-config --cflags --libs gtk+-2.0`” ，把执行得到的结果中以“-I”开头的添加到 Include 配置的 Include Paths(-I)，以“-L”开头的添加到 Library 配置的 Library Paths(-L)， 以“-l”开头的添加到 Library 配置的 Library Files(-l)，其它的添加到基本配置的其它编译选项中。

一个项目有多个 main 函数的源文件的情况

见项目配置的“主文件依赖”

编译程序

编译目标包含：all(编译所有可执行文件)、可执行文件（含main 函数的源文件去掉后

缀名得到）、clean（删除上次编译时产生的.o、.d 和可执行文件）。
编译方法：
方法1：右击“项目浏览器”中的项目，选择【编译】->选择要编译的目标。
方法2：在菜单栏选择【编译运行】->【设置当前项目】->选择要编译的项目， 菜单栏【编译】->选择要编译的目标。
方法3：在编译运行工具栏中选择要编译的项目和与之对应的编译的目标，点击【编译】按钮。
编译时会产生一个编译输出窗口，并将编译的输出显示在该窗口。

运行程序

运行程序包含所有由该项目的main 源文件编译产生的可执行文件。
运行方法：
方法1：右击“项目浏览器”中的项目，选择【运行】->选择要编译的目标。
方法2：在菜单栏选择【编译运行】->【设置当前项目】->【要运行的项目】， 菜单栏【编译运行】->【运行】->【要运行的程序】。
方法3：【编译运行工具栏】，在项目列表中选择要编译的项目，在程序列表中选择要运行的程序，在程序运行参数编辑栏输入程序参数点击【运行按钮】。
点击运行之后将弹出一个控制台窗口执行该程序。
只有方法 3 才可以给程序运行添加参数。