

vcHelper 操作手册

Hanford

2016 年 11 月 28 日

目 录

第 1 章 安装	1
1.1 功能简介	1
1.2 安装	1
1.3 配置	1
第 2 章 路径转换	3
2.1 长/短路径	3
2.2 绝对/相对路径	3
第 3 章 #include"相对路径"	5
第 4 章 #include"*.*"	10
第 5 章 修改项目文件	13
第 6 章 源文件树	15
6.1 添加一个目录内的源文件	15
6.2 复制文件树	16
第 7 章 编码与换行	18
第 8 章 文件重命名	21

第 1 章 安装

1.1 功能简介

在使用 VC++的时候，有些问题处理起来是比较繁琐的，如：

1、`#include` 语句以及项目设置里，经常会用到相对路径。完全靠人工来确定相对路径是比较繁琐的一件事情；

2、对一个项目的目录结构进行了调整之后，源文件、头文件要重新加入项目里，否则可能会打不开。而且很多`#include` 语句也要更改；

3、一份代码由 VC++6.0、VC++2010 共享，在 VC++6.0 IDE 里增加、删除了若干文件后，如何同步 VC++2010 的项目文件，保证 VC++6.0 和 VC++2010 编辑、编译的都是相同的文件？

vcHelper 程序就是专门用来解决这些问题的。

1.2 安装

vcHelper 是绿色软件，没有安装包，也不会往注册表里写任何数据。它只有如下几个文件：

表 1.1 文件说明

文 件	说 明
vcHelper.exe	Windows XP 以上的操作系统，请运行本程序
vcHelperA.exe	Windows XP 以下的操作系统，请运行本程序
Config.txt	配置文件
Version.txt	版本信息

1.3 配置

vcHelper 有时会调用 UltraEdit。如果 UltraEdit 的安装版本是 14.00a+1，那么 vcHelper 会自动找到 UltraEdit 的安装路径。如果 UltraEdit 的安装版本不是 1

4.00a+1, 请在 Config.txt 里设置 UltraEdit 的安装路径, 如下图所示:

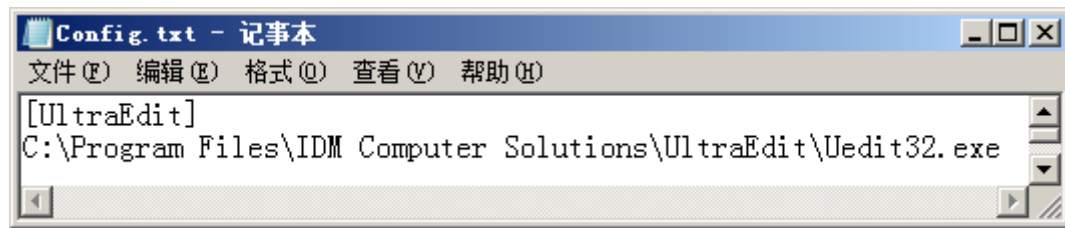


图 1.1 UltraEdit 安装目录

第 2 章 路径转换

2.1 长/短路径

如下图所示，本程序可进行长/短路径的相互转换：



图 2.1 长/短路径

“绝对路径”的文本框内输入路径，单击“短路径”按钮，将把长路径转换为短路径；单击“长路径”按钮，将把短路径转换为长路径。

注意：

- 1、路径应该是存在的，否则无法转换；
- 2、GetShortPathName 有时无法正常工作。如：在笔者的电脑上，操作系统为 64 位 Windows 7 旗舰版，GetShortPathName 无法将 C:\Program Files 转换为短路径。

2.2 绝对/相对路径

#include 语句有时需要用到相对路径。如：在源文件 C:\VC\Func.cpp 里要

包含头文件 C:\Share\String\Comp.h。可以用如下语句：

```
#include "..\\Share\\String\\Comp.h"
```

..\Share\String\Comp.h 就是 C:\Share\String\Comp.h 相对于 Func.cpp 所在目录 C:\VC 的相对路径。

使用两个反斜杠比较麻烦，可以用一个斜杠来代替。编译时，VC++会自动将斜杠转换为反斜杠。

```
#include "../Share/String/Comp.h"
```

这两个文件的相对路径还是比较简单的，假如 Func.cpp 在 C:\VC\MFC\VC6\ProjA，这个相对路径是什么呢？有些复杂了吧？交给 vcHelper 来完成该工作吧。下图中，在“基准目录/文件”里输入“C:\VC\MFC\VC6\ProjA”，在“绝对路径”里输入“C:\Share\String\Comp.h”，单击“>>”按钮即可计算出相对路径并显示出来。单击“<<”按钮可以进行逆转换。

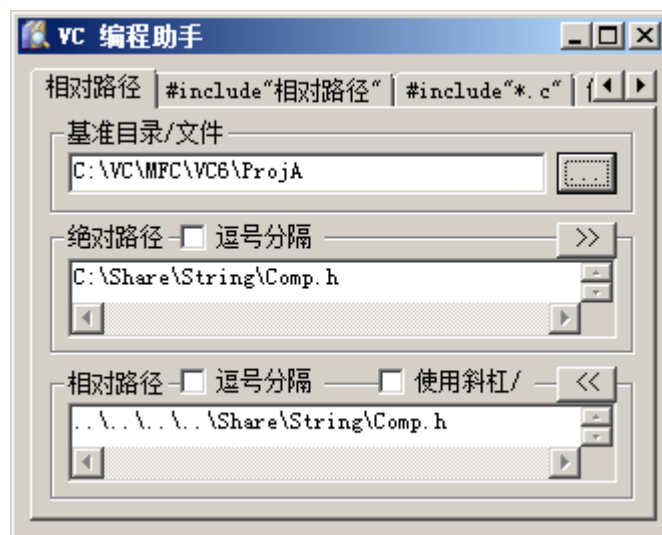


图 2.2 绝对/相对路径

说明：

- 1、程序会自动判断“基准目录/文件”。如果它存在且是文件，程序会自动将文件名去掉，仅保留目录；
- 2、绝对路径和相对路径可以有多个，一般一行代表一个路径。如果勾选“逗号分隔”则多个路径之间以逗号分隔；
- 3、勾选“使用斜杠/”，程序将把反斜杠\替换为斜杠/；
- 4、转换前，绝对路径、相对路径应统一使用长路径名。

第3章 #include"相对路径"

假定某个 VC 项目位于 C:\ProjA，该项目的许多文件都包含了 C:\Share 里的文件。如：`#include "../Share/Comp.h"`。现在因为项目的需要，将 ProjA 移动到了 C:\VC。即该 VC 项目目录变成了 C:\VC\ProjA。显然要将 `#include "../Share/Comp.h"` 改为 `#include "../../Share/Comp.h"`。这样的 `#include` 就几条也就罢了，如果有几十条甚至上百条，改起来就相当繁琐了，而且还可能会出错。

vcHelper 可以协助程序员完成这项工作，其操作步骤为：

1、设置“源文件目录”

如下图所示

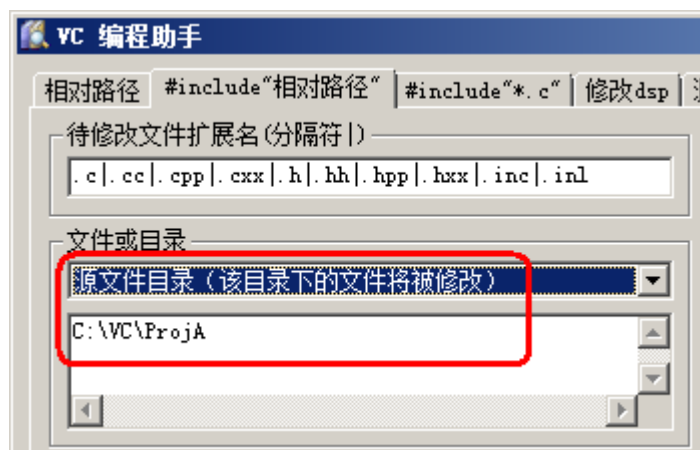


图 3.1

注意：源文件目录允许有多个，一行代表一个。

2、设置“引用目录”

如下图所示

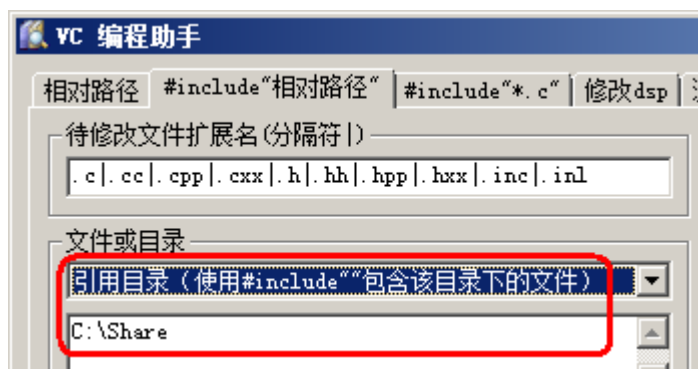


图 3.2

注意：引用目录允许有多个，一行代表一个。

3、设置“系统目录”

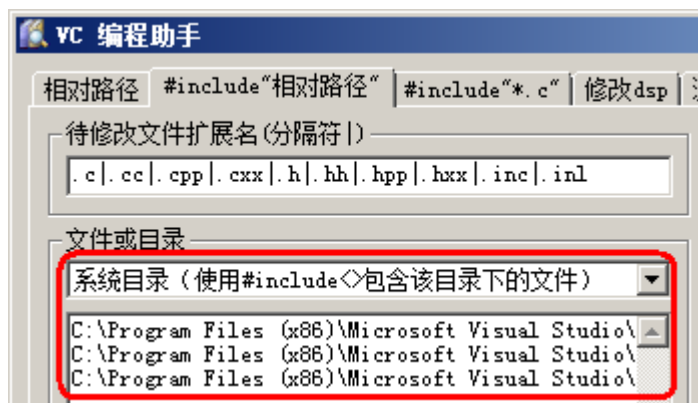


图 3.3

注意：系统目录允许有多个，一行代表一个。

4、设置修改选项

如下图所示：

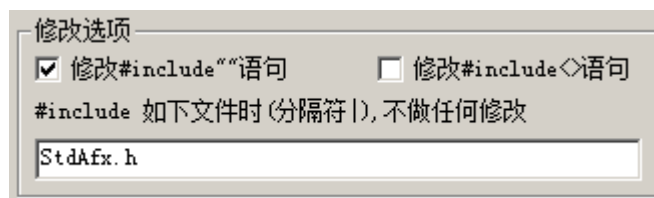


图 3.4

复选框“修改#include""语句”表示是否修改形如#include "a.h"的语句，一般情况下请勾中它。如果使用预编译头文件 StdAfx.h，那么语句#include "Std

Afx.h"就不应该被修改，所以上图的文本框中输入了 StdAfx.h;

复选框“修改#include<>语句”表示是否修改形如#include <stdio.h>的语句，一般情况下不用勾中它。

5、扫描文件

如下图所示，请单击“扫描”按钮：

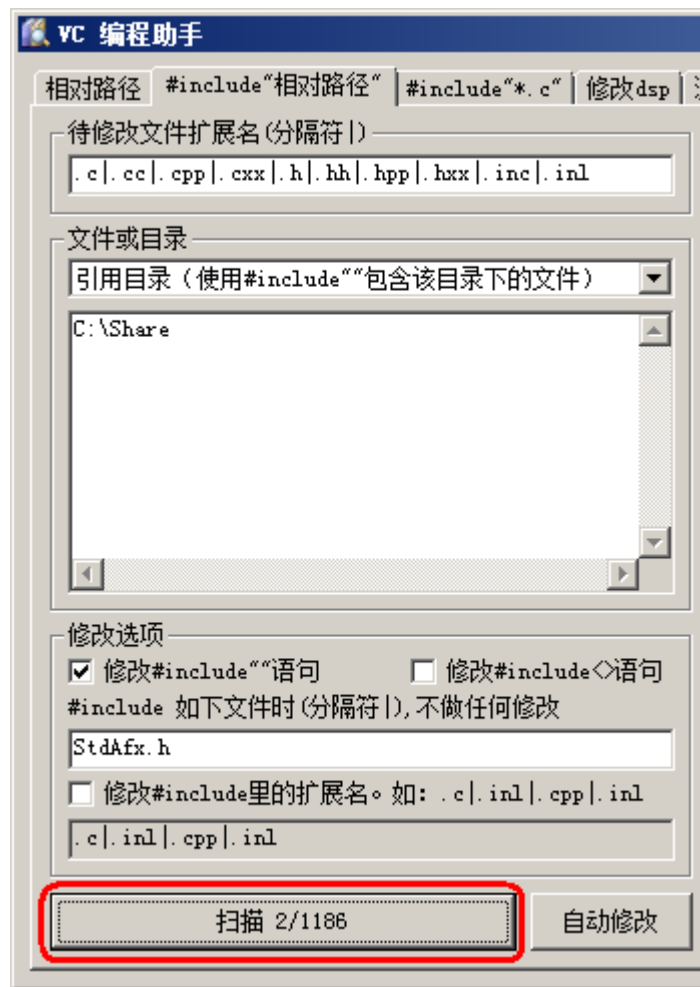


图 3.5

上图表示扫描到 1186 个文件，需要修改的文件有 2 个。

6、自动修改

如下图所示，单击“自动修改”按钮，将自动修改#include 语句

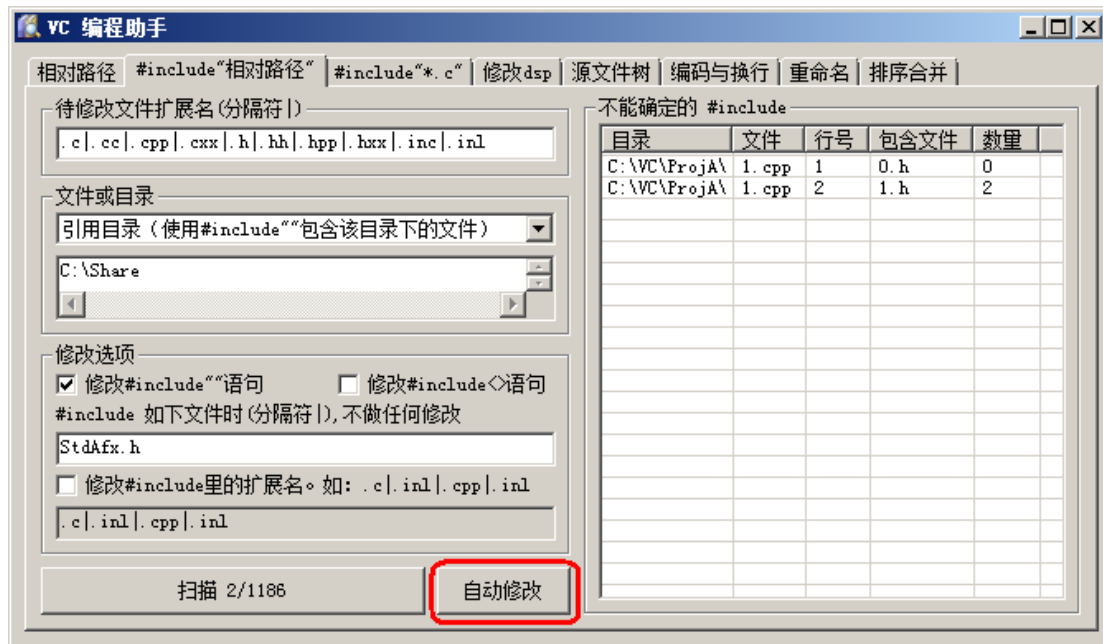


图 3.6

右侧的列表将列出不能确定的#include 语句。其内容被导出至文件 include.csv 里，可使用 Excel 打开、查看。

单击上图右侧列表的“数量”列标题，列表将按照数量进行排序。

数量为零，说明包含文件不存在。如上图中的 0.h 是不存在的，所以也就无法确定相对路径，也无法修改#include 语句；

数量大于 1，说明包含文件不唯一。如上图中的 1.h 有两个，vcHelper 无法确定究竟要包含哪个？此时，请使用鼠标左键双击该行，将显示如下界面：

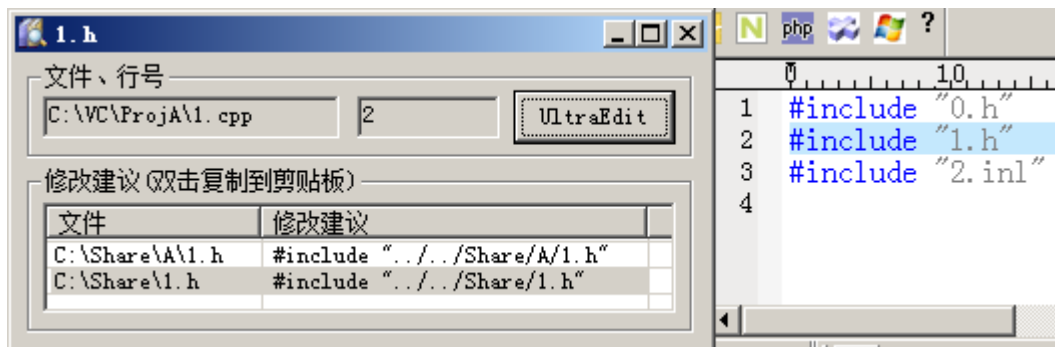


图 3.7

单击上图的按钮“UltraEdit”，程序将调用 UltraEdit 打开文件 C:\VC\ProjA\

1.cpp，并将文本插入符定位到第2行。上图右侧就是 UltraEdit 的界面。

上图说明：C:\VC\ProjA\1.cpp 的第2行是#include "1.h"。该语句包含的 1.h 有两个 C:\Share\A\1.h 和 C:\Share\1.h。请在列表中选择合适的文件，如第二行是正确的，请使用鼠标左键双击第二行第二列，然后在 UltraEdit 里选中第2行，按下 Ctrl+V 即可将原语句替换为#include "../Share/1.h"。

第 4 章 #include "*.c"

GSL (GNU Scientific Library) 里有一些 *.c 文件比较特殊：编译器并不直接编译它，而是某些 c 文件（宿主文件）嵌入了其内容，当编译宿主文件时这类文件才发生作用。

举例说明，G:\gsl-1.16\block\block.c 的部分内容如下：

```
#define BASE_DOUBLE
#include "templates_on.h"
#include "block_source.c"
#include "templates_off.h"
#undef BASE_DOUBLE

#define BASE_FLOAT
#include "templates_on.h"
#include "block_source.c"
#include "templates_off.h"
#undef BASE_FLOAT
```

可以看到：block_source.c 被多次嵌入到 block.c 里。编译器并不需要直接编译 block_source.c，只是在编译 block.c 时 block_source.c 才发挥作用。

GSL 这么做的目的是为了节省人工编写代码的工作量，上述代码的用意为：针对 double 编译一次 block_source.c，再针对 float 编译一次 block_source.c……这个非常类似于 C++ 的模板。

现在的问题是：对于一个并不想深入研究 GSL 代码的程序员而言，他根本无法区分 block_source.c 和 block.c：同样的 *.c 文件，如何确定哪个文件应该编译，哪个文件不应该编译？解决方法就是重命名：把 #include "block_source.c" 修改为 #include "block_source.inl"，同时重命名 block_source.c 为 block_source.inl。

使用 vcHelper 可自动完成此项工作，具体操作如下：

运行 VS2008，打开“查找和替换”对话框，并按下图进行配置，然后单击“查找全部”按钮。

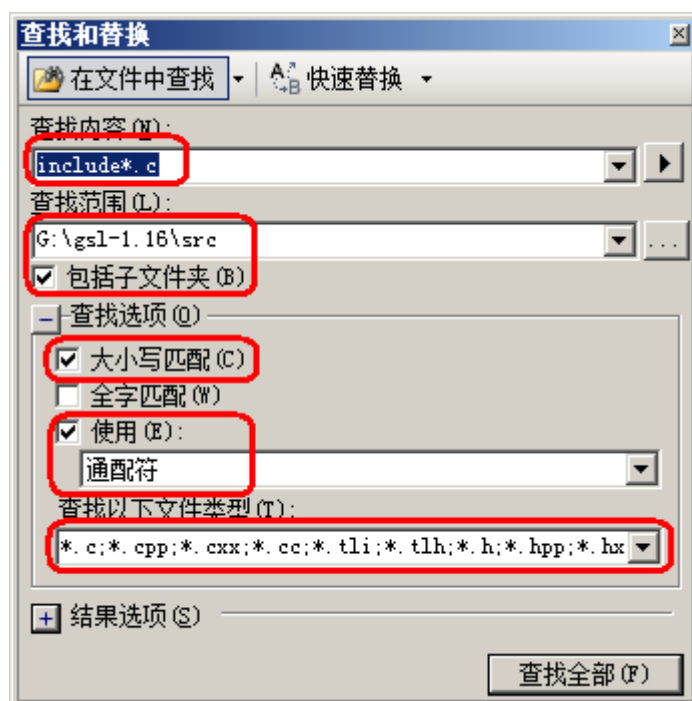


图 4.1

查找结果如下表所示。

查找全部 "include*.c", 大小写匹配, 通配符, 子文件夹,		
G:\gsl-1.16\src\block\block.c(7):#include "block_source.c"		
G:\gsl-1.16\src\block\block.c(13):#include "block_source.c"		
.....
.....
G:\gsl-1.16\src\vector\view.c(174):#include "view_source.c"		
匹配行: 956	匹配文件: 161	合计搜索文件: 1344

通过上表可知共有 956 个匹配项，手动修改的话工作量会比较大，而且容易出错。因此，可以通过编写程序来实现修改，具体而言就是：替换 G:\gsl-1.16\src\block\block.c 里的第 7 行代码 #include "block_source.c" 为 #include "block_source.inl"，同时重命名 G:\gsl-1.16\src\block\block_source.c 为 block_source.inl。上表中的 956 项需要逐一进行修改。

运行 vcHelper 程序，然后把上表内容复制到 “#include "*.c” 页面的文本框内，最后单击 “应用” 按钮，完成相应的更改工作。如下图所示：

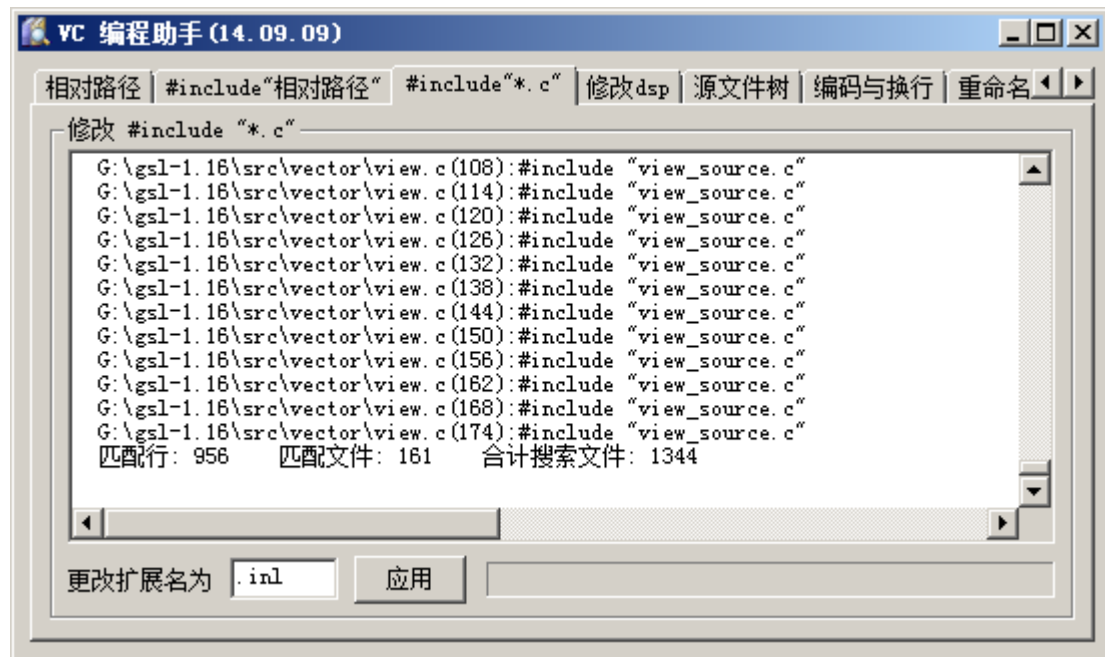


图 4.2

第 5 章 修改项目文件

VC++6.0 的项目文件扩展名为 `dsp`。EVC 的项目文件扩展名为 `vcp`。这两种文件的格式非常相似。VC++7.0 以后的项目文件扩展名为 `vcproj`，它是一个 xml 文件，与 `dsp`、`vcp` 的格式差别较大。

假定现在有一个 VC++6.0 的项目 `Test`。它有三个文件：`Test.h`、`Test.cpp`、`Test.dsp`。这三个文件原来是在同一个目录下的，而且项目可以正常编译。现在将 `Test.h` 移到 `Inc` 目录下，将 `Test.cpp` 移到 `Src` 目录下，项目还能正常编译吗？

使用记事本打开 `Test.dsp`，可以看到如下两段文本：

表 4.1 `Test.dsp` 文件内容

文件内容	说 明
# Begin Source File SOURCE=.\Test.cpp # End Source File	dsp 文件对 <code>Test.cpp</code> 的引用 显然 <code>.\Test.cpp</code> 应该更改为 <code>.\Src\Test.cpp</code>
# Begin Source File SOURCE=.\Test.h # End Source File	dsp 文件对 <code>Test.h</code> 的引用 显然 <code>.\Test.h</code> 应该更改为 <code>.\Inc\Test.h</code>

移动 `Test.h` 和 `Test.cpp` 之后，`Test.dsp` 文件对这两个文件的引用就无效了。此时将无法编译 `Test` 项目。只能在 IDE 里删除 `Test.h` 和 `Test.cpp`，然后再重新添加。或者修改 `dsp` 文件。这也是一件繁琐的事情。

`vcHelper` 可以协助程序员完成这项工作，其操作步骤可以参考下图。

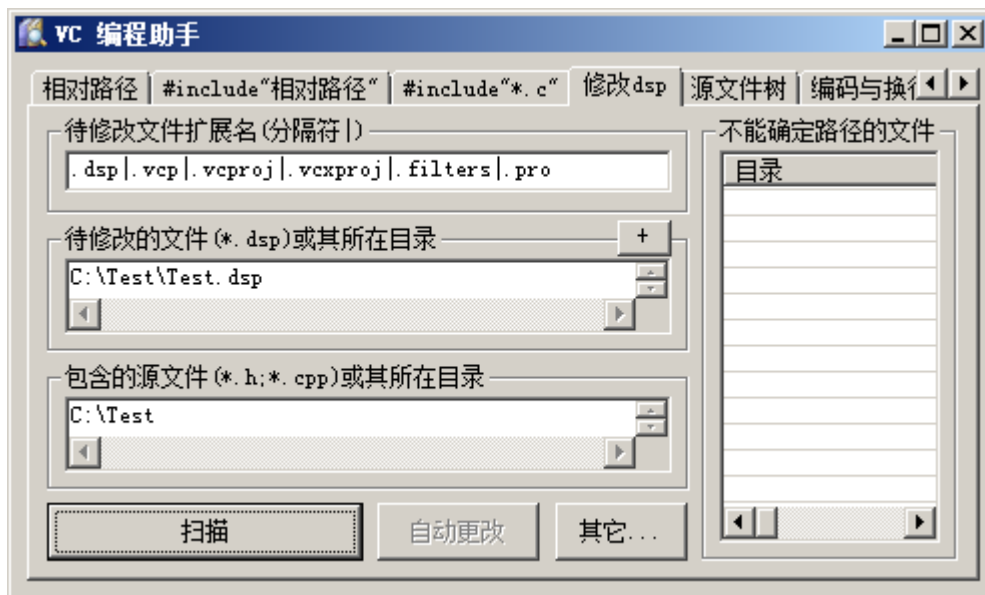


图 5.1

设置好文件、目录后，单击“扫描”、“自动更改”按钮即可。

第6章 源文件树

6.1 添加一个目录内的源文件

把 G:\gsl-1.16 目录下一千多个文件 (*.c;*.h;*.inl) 添加到 VC 项目里，这是一项艰巨的工程。这里使用 vcHelper 来添加，如下图所示：



图 6.1

“生成文件树”里的文本框输入 G:\gsl-1.16\src，然后单击“生成文件树”按钮，即可生成文件树。上图可以看到：生成的文件树里有 1126 个文件。

拖动 G:\gsl-1.16\make-dll、G:\gsl-1.16\make-libS、G:\gsl-1.16\make-libT、G:\gsl-1.16\make-libD 目录到“替换文件树”里的文本框，则 vcHelper 会自动找到这些目录里的 VC 项目文件 (*.dsp;*.vcproj;*.vcxproj)。单击“替换文件树”按钮，则 vcHelper 会把 VC 项目文件里的文件树替换为生成的文件树。如下图所示：

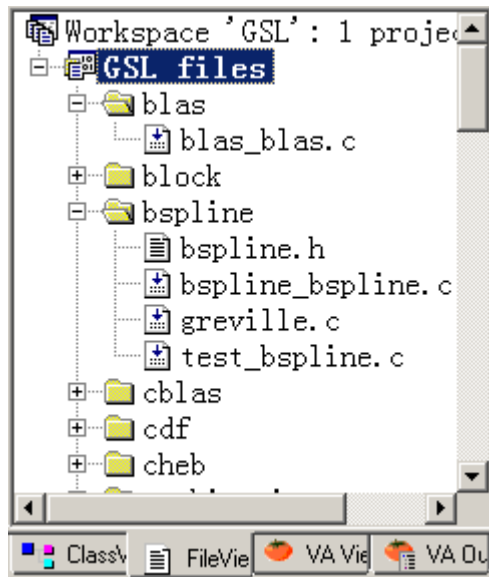


图 6.2

6.2 复制文件树

下图将 GSL.dsp（vc6 项目文件）中的文件树复制到 GSL.vcproj（vc2005 和 vc2008 项目文件）和 GSL.vcxproj（vc2010 项目文件）中。

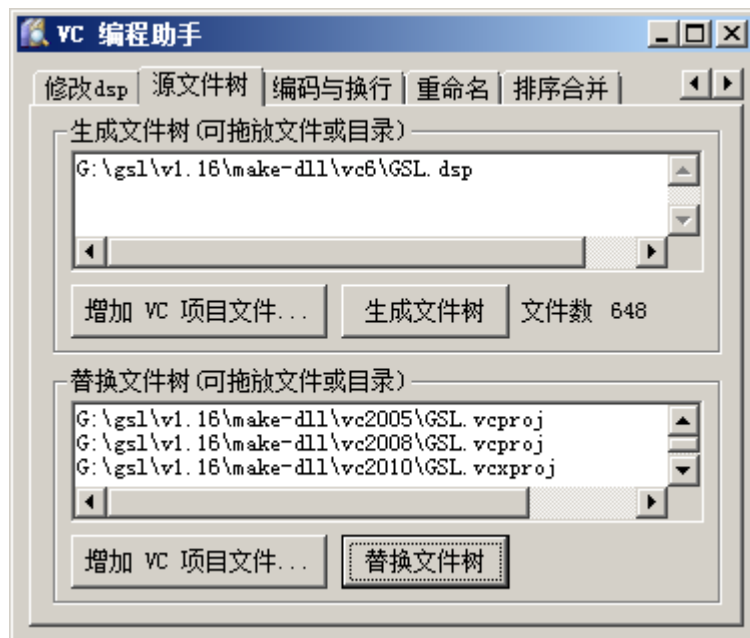


图 6.3

上图中，单击“生成文件树”按钮，程序将读取 `GSL.dsp` 中的文件树。再单击“替换文件树”按钮，程序将替换 `GSL.vcproj` 和 `GSL.vcxproj` 里的文件树。

第 7 章 编码与换行

目前，源代码文件（文本文件）的编码呈百花齐放的状态。如：VC++6.0 仅支持 ANSI 编码；Qt、Eclipse 最好用带有 BOM 的 UTF-8 编码；Android Studio 最好用不带 BOM 的 UTF-8 编码……

还有换行符，Max、Unix、Windows 三者并不统一。网上下载下来的 Linux 开源代码，在 Windows 的记事本里显示为一行。

可使用本程序改变文本文件的编码及换行符。如：GSL（GNU Scientific Library）源代码的行结束符是换行（0AH），现在将其更改为回车、换行（0DH、0AH）。操作步骤如下：

Windows 资源管理器里，进入 G:\gsl-1.16\src 搜索 *.c 和 *.h，如下图所示。

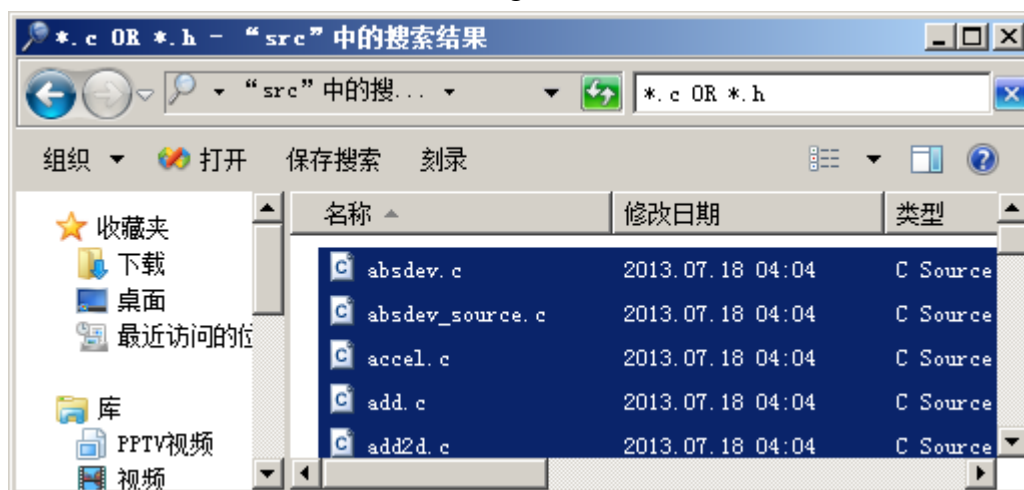


图 7.1

把上图中的所有文件拖放到程序 vcHelper 的“编码与换行”页面，即可完成编码、换行符的转换。如下图所示。或者上图中 Ctrl+C 复制文件，下图中 Ctrl+V 粘贴文件，也可完成编码、换行符的转换。

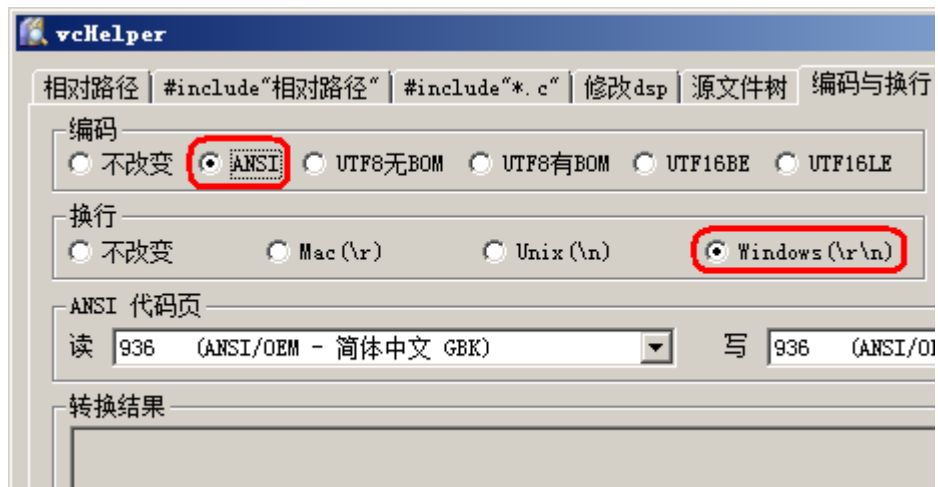



图 7.2

上图中，编码选择的是 ANSI，这是为了兼容 VC++6.0；换行选择的是\r\n。vcHelper 对某一个文件的处理：

- 1、将文件内容载入内存，其实就是一个字符串；
- 2、将内存字符串的编码转换为 UTF-8。对于 ANSI 编码，请指定“ANSI 代码页”下的“读”；
- 3、对于 ANSI 编码的字符串，程序会检测它是否为 UTF-8 编码。如果程序无法区分字符串的编码，会显示如下界面：



图 7.3

因为 GBK 编码的“联通”也有可能是带 BOM 的 UTF-8 编码“”。程序无法区分，所以显示上图所示界面，供用户选择。显然，这里应该单击“选择 ANSI”按钮。

对于不带 BOM 的 UTF-8 编码，也可能需要选择代码页，如下图所示：

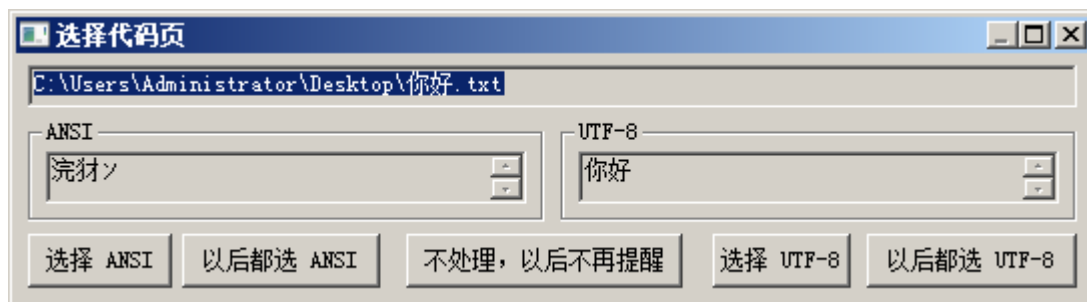


图 7.4

UTF-8 编码的“你好”，可以被 GBK 编码理解为“浣豺ソ”。上图中，应该单击“选择 UTF-8”按钮。

如果确定所有的文本文件都是 ANSI 或 UTF-8 编码，就请单击“以后都选 ANSI”或“以后都选 UTF-8”按钮。处理后继文件时，将不再显示上图所示的界面进行询问。

如果不确定怎么处理，请单击“不处理，以后不再提醒”按钮或关闭上图所示窗口，则程序对该文件不会进行处理。

3、进行换行处理，将换行符替换为设定值；

4、改变编码为设定值，然后写入文件。如果写入文件时的编码是 ANSI，请指定图 7.2 中“ANSI 代码页”下的“写”。

对“ANSI 代码页”的设置要特别仔细，一旦设置错误就会出现乱码。下图的设置，用于将 GB18030 编码转换为 GBK 编码。

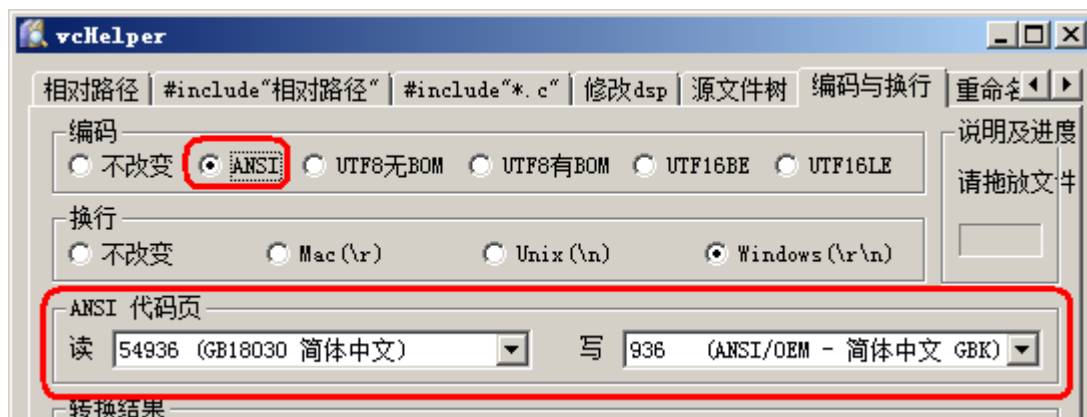


图 7.5

第8章 文件重命名

GSL (GNU Scientific Library) 的源代码文件 (*.c) 存在同名的现象, 如下图所示: 在 G:\gsl-1.16\src 目录查找 *.c, 将会发现有多个 inline.c 文件。



图 8.1

VC++编译时不允许源文件同名, 为此可将上图所有的 *.c 文件拖放至下图所示的“重命名”页面即可完成文件重命名。

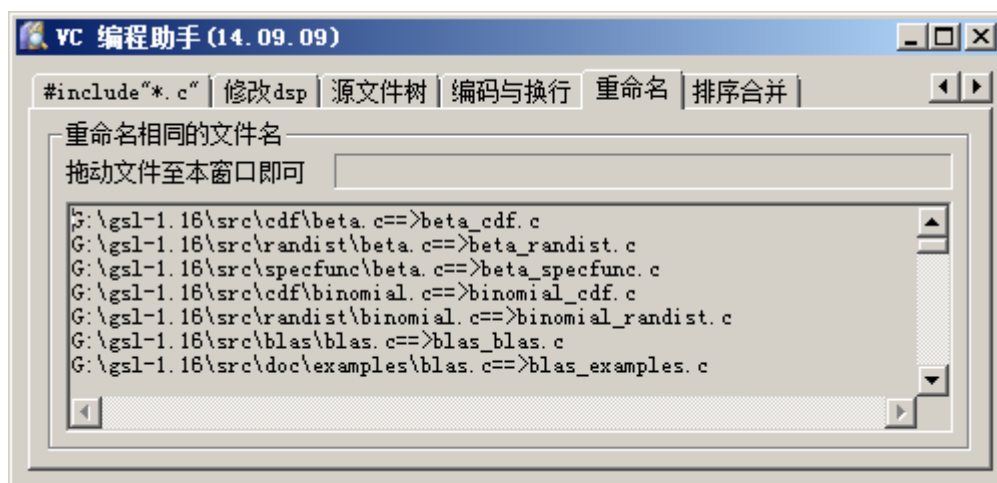


图 8.2