作为一个开放源代码的操作系统，Linux 附带的源代码库使得广大爱好者有了一个广泛学习、深入钻研的机会，特别是 Linux 内核的组织极为复杂，同时，又不能像 windows 平台的程序一样，可以使用集成开发环境通过察看变量和函数，甚至设置断点、单步运行、调试等手段来弄清楚整个程序的组织结构，使得 Linux 内核源代码的阅读变得尤为困难。

当然 Linux 下的 vim 和 emacs 编辑程序并不是没有提供 变量、函数搜索，彩色显示程序语句等功能。它们的功能是非常强大的。比如，vim 和 emacs 就各自内嵌了一个标记程序，分别叫做 ctag 和 etag，通 过配置这两个程序，也可以实现功能强大的函数变量搜索功能，但是由于其配置复杂，linux 附带的有关资料也不是很详细，而且，即使建立好标记库，要实现 代码彩色显示功能，仍然需要进一步的配置（在另一片文章，我将会讲述如何配置这些功能），同时，对于大多数爱好者来说，可能还不能熟练使用 vim 和 emacs 那些功能比较强大的命令和快捷键。

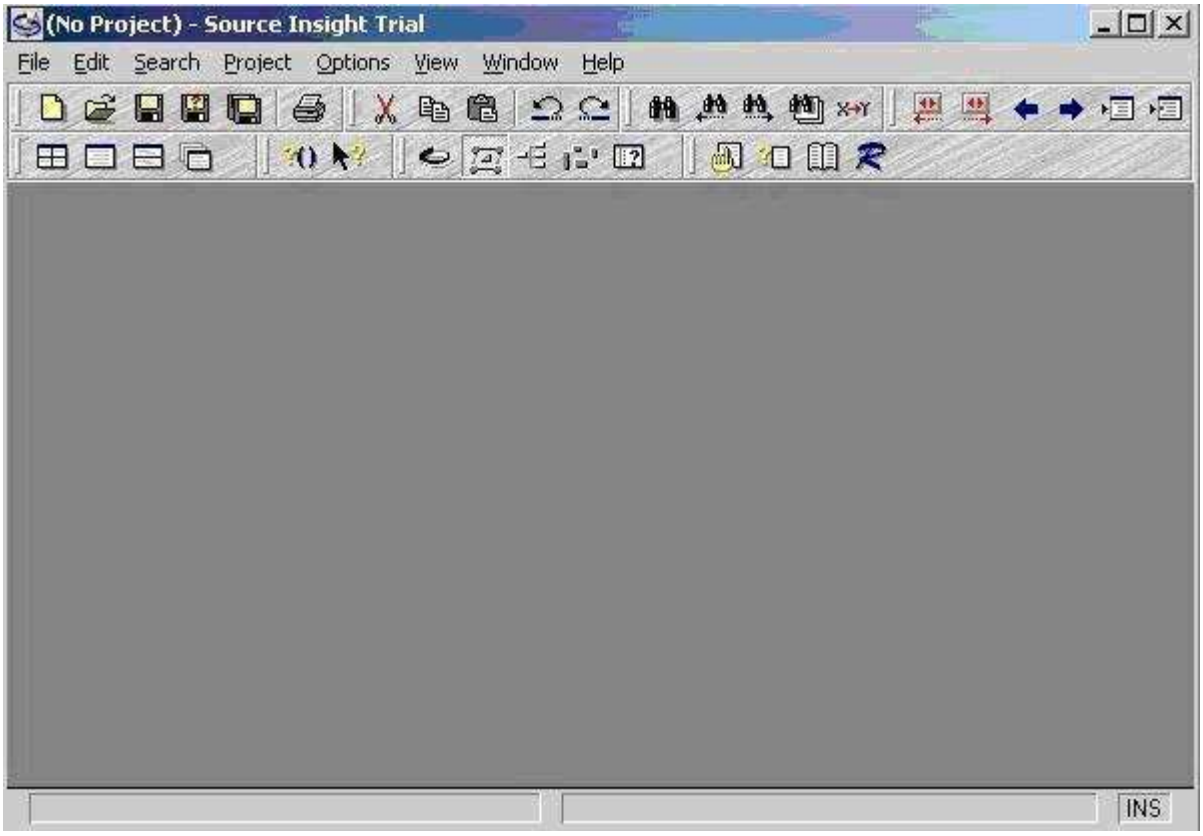
为了方便的学习 Linux 源程序，我们不妨回到我们熟悉的 window 环境下，也算是“师以长夷以 制夷”吧。但是在 Window 平台上，使用一些常见的集成开发环境，效果也不是很理想，比如难以将所有的文件加进去，查找速度缓慢，对于非 Windows 平台的函数不能彩色显示。于是笔者通过在互联网上搜索，终于找到了一个强大的源代码编辑器，它的卓越性能使得学习 Linux 内核源代码的难度大大降低，这便是 Source Insight3.0，它是一个 Windows 平台下的共享软件，由

于 Source Insight 是一个 Windows 平台的应用软件，所以首先要通过相应手段把 Linux 系统上的程序源代码弄到 Windows 平台下，这一点可以通过在 linux 平台上将/usr/src 目录下的文件拷贝到 Windows 平台的分区上，或者从网上光盘直接拷贝文件到 Windows 平台的分区来实现。

下面主要讲解如何使用 Source Insight，考虑到阅读源程序的爱好者都有相当的软件使用水平，本文对于一些琐碎、人所共知的细节略过不提，仅介绍一些主要内容，以便大家能够很快熟练使用本软件，减少摸索的过程。

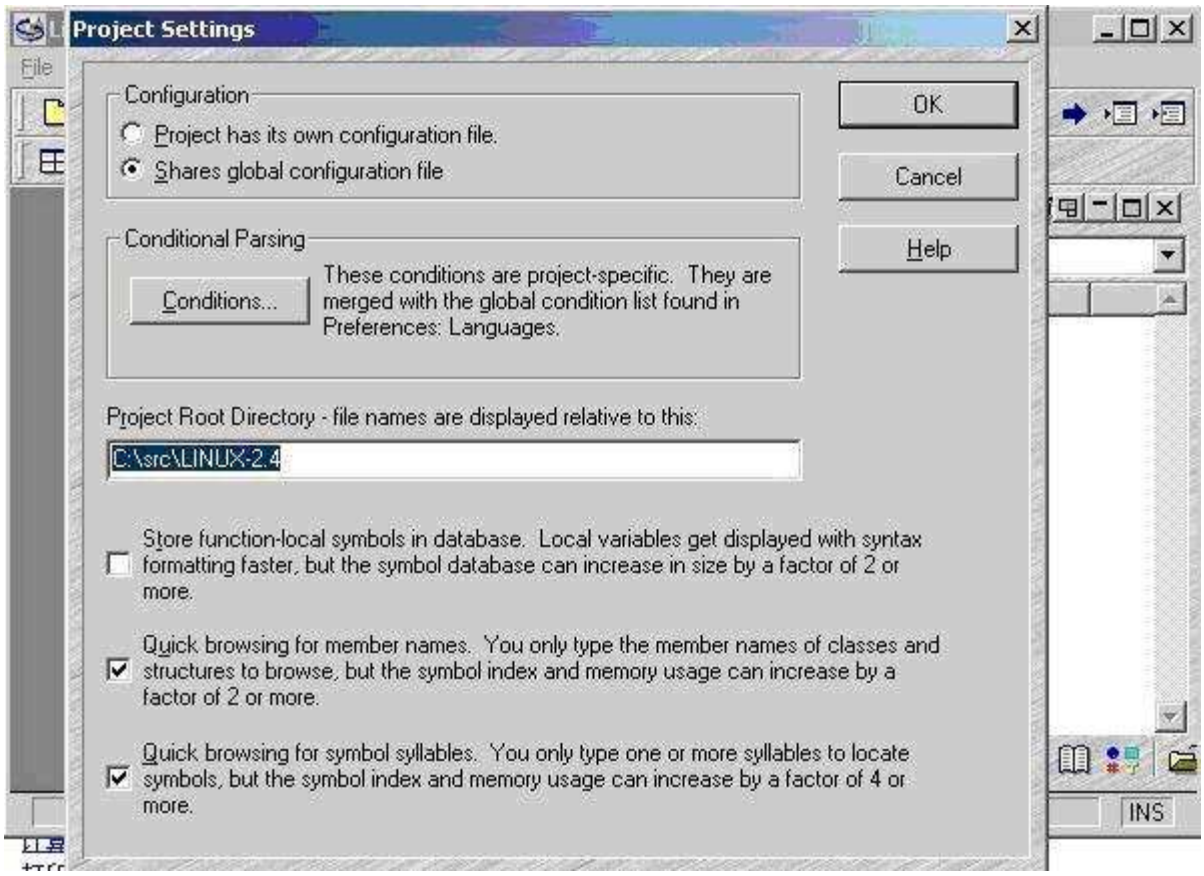
安 装 Source Insight 并启动程序，可以进入图 1 界面。在工具条上有几个值得注意的地方，如图所示，图中内凹左边的是工程按钮，用于显示工程窗口的情况；右边的那 个按钮按下去将会显示一个窗口，里边提供光标所在的函数体内对其他函数的调用图，通过点击该窗体里那些函数就可以进入该函数所在的地方。

图 1 Source Insight 界面图



由于 Source Insight 实质上是一个支持多种开发语言（java,c ,c++等等）的编辑器，只不过由于其查找、定位、彩色显示等功能的强大，而被我们当成源代码阅读工具使用。所以，为了有效的阅读源程序，首先必须选择功 能菜单上的 “Project”选项的子菜单“New Project”新建一个项目，项目名称可以自由选定，当然也可以选择删除（Remove）一个项目。当删除一个项目的时候，并不删除原有的源代码文件，只是将该软件生成的那些工程辅助文件删除。设定之后，将会弹出一个对话框如图 2，接受默认选择，如果，硬盘空间足够，可以将第一个复选框选上，该选项将会 需要与源代码大致同等的空间来建立一个本地数据库以加快查找的速度。

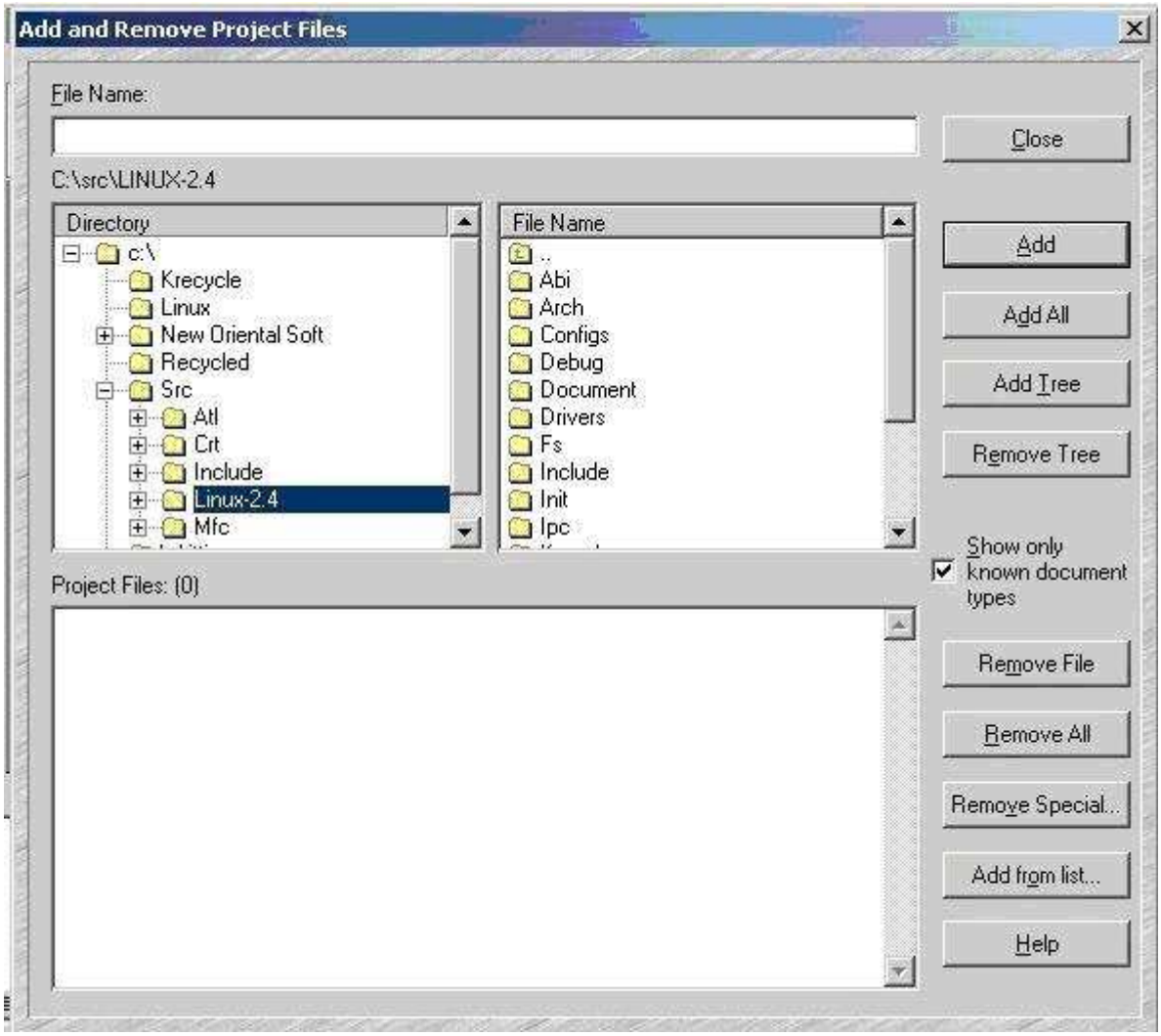
图 2 工程设置



点击“OK”按钮，接受选择后，将会有一个新的对话框弹出，在这个对话框里，可以选择将要阅读的文件加入工程，一种方式是通过在 File Name 中输入要阅读源代码文件的名称，点击“Add”按钮将其加入，也可以通过其中“Add All”和“Add Tree”两个按钮可以将选中目录的所有文件加入到工程中，其中“Add All”选项会提示加入顶层文件和递归加入所有文件两种方式，而“Add Tree”相当于“Add All”选项的递归加入所有文件，可以根据需要使用，就我来说，更喜欢“Add Tree”一些。由于该程序采用了部分打开文件的方式，没有用到的文件不会打开，所以，加入数千个文件也不用担心加

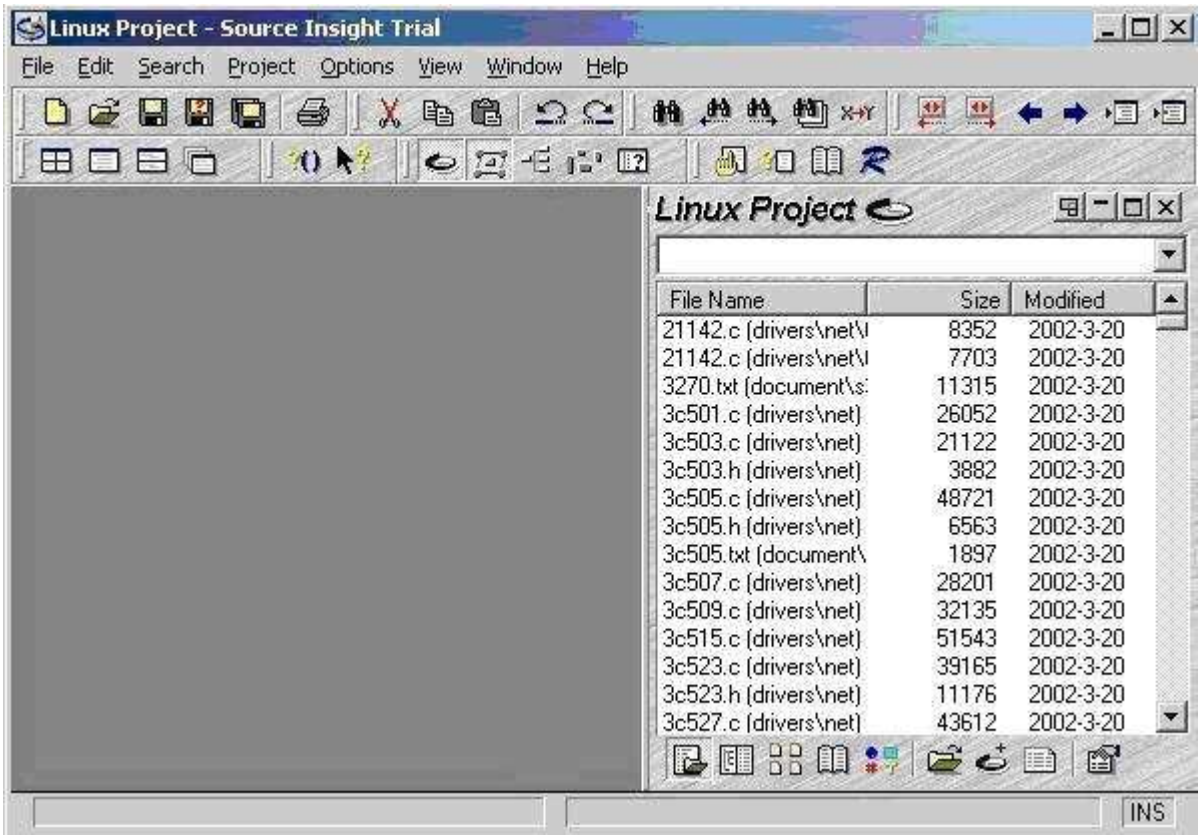
入的文件超出程序的所能容忍的最大值， 我就是采用“Add Tree”的方式将 Linux2.4 内核的四千五百九十一个文件加入的。

图 3 添加文件



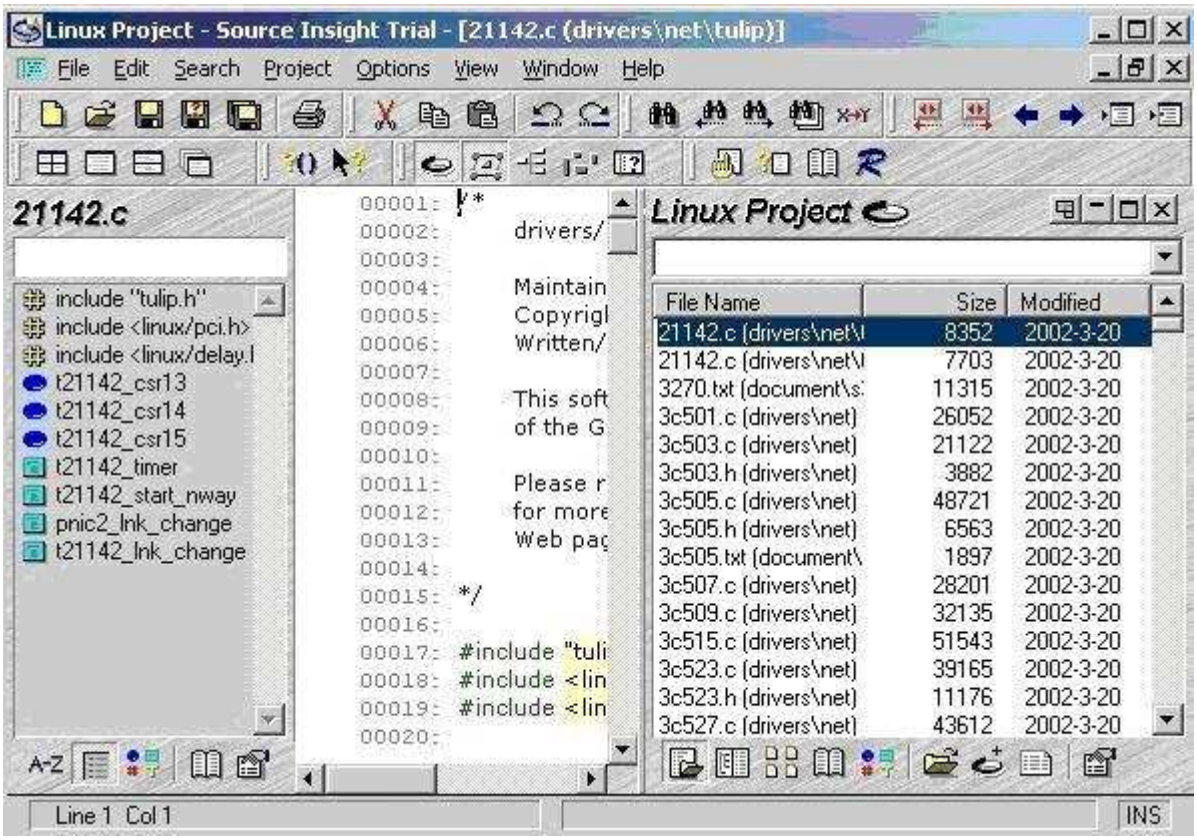
加入文件后，点击一个文件，可以出现使用界面，如图 4 所示，其中，右边的那个窗口（Linux Project，即工程窗口）缺省按照字母顺序列出当前工程中所有的文件。

图 4 工作窗口



点击一个文件就可以打开该文件，显示如图 5 所示，进入到右边的那个窗口分别可以以文件列表的方式，列出所有的文件，每个窗体下边有一排按钮，左边的窗口 （21142.c）从左至右分别为：按字母顺序排列所有标记、按照文件中行数顺序排列标记、按照类型排列标记、浏览本地文件标记、标记窗口属性。右边的窗 口（Linux Project）从左至右分别为：按字母顺序文件列表、显示文件夹、按照文件类型归类文件、全部文件的所有标记列表、按照标记类型归类标记、跳转到定义 处、显示标记信息、浏览工程标记、查找函数调用、工程属性，其中全部文件的所有标记列表选项可能要一段时间抽取标记，同步到数据库去，如果开始选择了建立 标记数据库，将会在今后节省同步时间，最有用的莫过于浏览标记信息和查找函数调用，前者可以

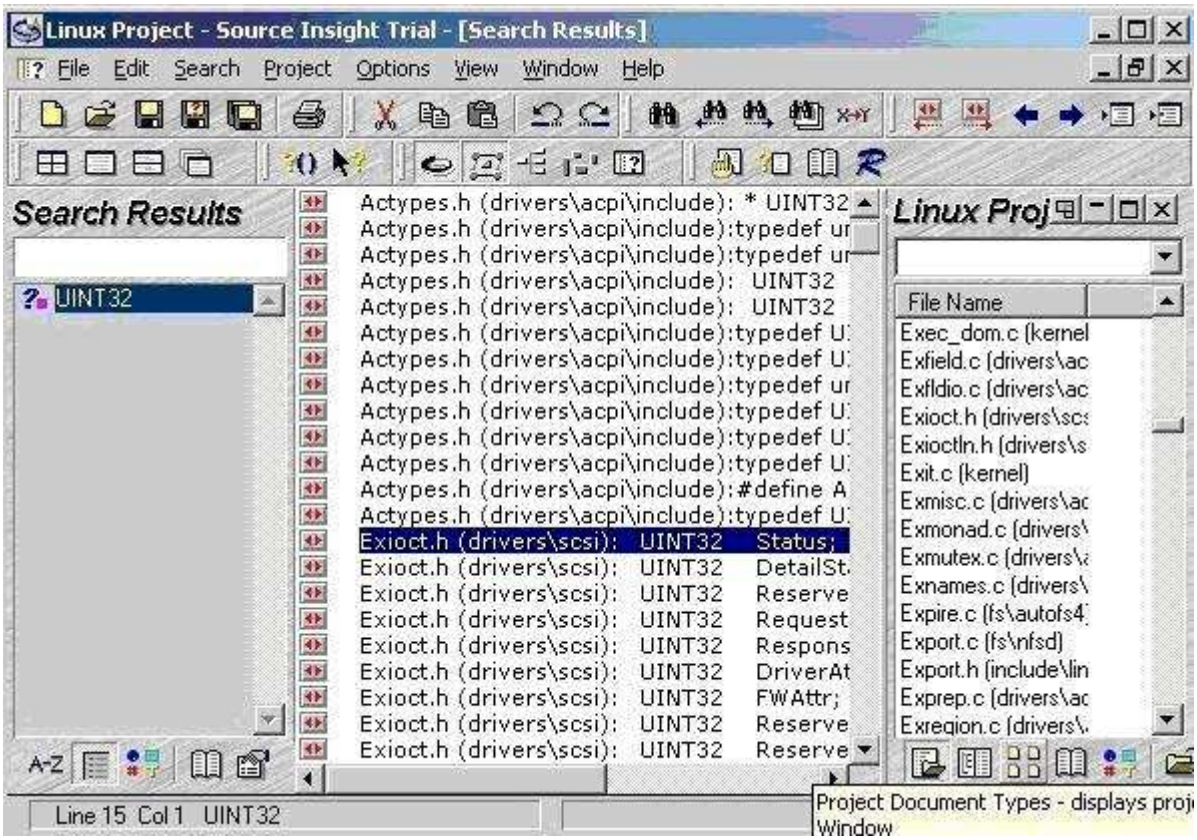
通过“Jump”按钮在不同的地方查找同样的标志，还可以通过“Reference”按钮结合后者进行全局的标记查找。



Reference 功能是 Source Insight 的特色之一，它可以在速度极快的在整个工程中找到所有的标记，并且在该行程序的前边加上红色箭头的小按钮链接上。图 6 是一个 Reference 搜索后的结果，它可以有两种模式，一种集中显示结果，图 6 显示的就是这种模式，在这种模式下，可以通过前边的红色箭头小按钮进入另外一 种模式，该标记的具体的所在处，也可以通过标记的具体所在处点击红色箭头小按钮进入警种模式，还可以通过工具条上的两个红色小箭头直接在第二种模式下前后 移动，察看相应信息。它的这个强大的功能使得阅读 Linux 源程序有如神助。但是要注意的是，当进行了第二次

“Reference”时，它会提示你将结果 集附加在第一个结果集的后边还是取代第一个结果集。如果选择前者，不能对结果集根据前后两次搜索结果进行分类，然后在其子类里进行移动，只能在整个结果集 里移动；如果，选择后者，结果集将会被替换为第二次搜索的结果，略微有些不方便。

图 6 Reference 的搜索结果



当然，Source Insight 还提供了一些其他常见的便利。比如:右键菜单几乎包含了程序的所有功能，可以在编辑窗口为程序加上行号，还可以统计整个工程的程序行数,当然还有功能强大 却用不上自动完成功能，似乎连它的 30 天试用期也是别有用心——可以迫使你尽可能快速的阅读源程序，其他一些技巧大家可以在使用过程中慢慢

摸索。怎么样？ 爱好读源代码的朋友，不妨马上去下载一个，去开始我们的 Linux 内核探险之旅吧！

SourceInsight 使用技巧

2008/05/19 16:44 [学习研究]

1、缩进与 tab

（1）Options 菜单→Preferences→Typing 卡，勾掉下面两项：

Typing tab indents line，regardless of selection，空行按 tab 无法前进

Typing tab replaces current selection，选定部分内容、再按 tab 时会清除所选

（2）Options 菜单→Document Options（针对不同文件类型，分别进行设置）→下拉左上文件类型框、选择合适类型（c 源文件）→Editing Options 框中，tab width=2→ Editing Options 框中，勾选 Expand tabs（这样，按 tab 键、等价于输入 2 个空格）

（3）Options 菜单→Document Options→选择合适的文件类型→点击右边中间的 Auto Indent 钮→在弹出的框中，左边一定要

点Smart，右边有两个复选框Indent Open Brace和Indent Close Brace，具体效果可以看SIS的HELP。按照部门里的编程风格要求，最方便的就是把两个复选框都取消掉，然后点OK。

勾选Auto Indent之SMART的效果：在C程序里，如果遇到行末没有分号的语句,如IF, WHILE, SWITCH等，写到该行末按回车，则新行自动相对上一行缩进两列。

勾掉Indent Open Brace和Indent Close Brace的效果：继上一段，在相对缩进行里，如果输入"}"，则自动和上一行列对齐（好像勾不勾都会有这个功能）；而输入"{"时，不会与下面的行对齐（这是勾上Indent Open Brace时的效果）。

2、向项目中添加文件时，只添加特定类型的文件（文件类型过滤器）

有个同事比较生猛，得整汇编代码，但在SIS里建立PROJECT并ADD TREE的时候，根据默认设置并不会把该TREE里面所有汇编文件都包含进来，只加了.inc和.asm后缀的，.s后缀的没有。而且用SIS打开.s的文件，一片黑白没有色彩，感觉回到DOS的EDIT时代了…… 解决方法是在**Options->Document Options**里面，点左上的**Document Type**下拉菜单，选择**x86 Asm Source File**，然后在右边的**File filter**里***.asm；*.inc；**的后面加上*.s；接着CLOSE就可以了。上面两个问题解决了，但注意加入*.s后还需要重新ADD TREE一遍才能把这些汇编加到PROJECT里面。

3、去掉功能强大但是无用的自动完成功能

Options 菜单→Preferences →Typing 卡→Auto Completion 框，勾掉 Use automatic symbol completion window（这里是 SIS 的全局设置）

Options 菜单→Document Options→Editing Options 框中，勾掉 Allow auto-complete（局部设置）

上面两项必须全部勾选，才能启用 Auto Completion 功能

4、恢复小键盘的″+，-，*，/″功能

Options 菜单→Key assignments，通过关键词 Scroll 找到 Scroll Half Page Up，取消小键盘/；通过关键词 Scroll 找到 Scroll Half Page Down 取消小键盘*；通过关键词 Function 找到 Function Up，取消小键盘-，通过关键词 Function 找到 Function down，取消小键盘+。

5、恢复 ctrl+a 的全选功能

通过关键词 save 找到 save all，更改为 ctrl+shift+a，通过关键词 select 找到 select all， 更改为 ctrl +a

6、解决字符等宽对齐问题。

SIS 默认字体是 VERDANA，很漂亮。这网页上应该也是用的 VERDANA 字体。但由于美观的缘故，VERDANA 字体是不等宽的。比如下面两行

lllllllll

MMMMMMMMM

同样 10 个字符，长度差多了.用 VERDANA 来看程序，有些本应该对齐的就歪了。解放方法是使用等宽的字体，但肯定比较丑。可以用 DOS 字体，也就是记事本里的默认字体 sysfixed 很丑，要有心理准备。比较推荐的是用 Courier New。

SourceInsight 提供的功能

1、解析日志信息时非常有用的 Source Link

总地说来，SourceLink 根据特定的搜索模式，把当前文件中满足模式的行、链接到由该行指定的其他源文件中。

所谓特定的搜索模式，共有两种″File, then line″和″Line, then file″，其中前后两部分依靠正则表达式的**组的概念**来予以分割。如果当前文件具有匹配行，比如″Error d:tcsrcq5.c 18: Lvalue required in function jsSort″，那么 SourceInsight 在该行创建 SourceLink、把该行链接到由该行指定的文件中（即 d:tcsrcq5.c，第 18 行）。

1.1 创建 SourceLink

运行 Search 菜单的 Parse Source Links…命令，在弹出的框中、选择搜索模式、并填入相应的正则表达式串，点 OK，SIS 就会解析当前文件，如果有匹配，就创建 SourceLink。

1.2 在解析日志信息时，使用 SourceLink

可以打开日志信息，运行 Parse Source Links 命令，日志中能够匹配模式的每一行（通常是含有错误信息的行）、就会被设置上一个 SourceLink

1.3 在解析自定义命令输出时，使用 SourceLink

首先勾选 Custom Command 中的″Parse Links in Output″，然后选择特定的搜索模式，最后填入合适的正则表达式。这样，Source Insight 把输出信息作为当前搜索用文件；并且，如果有匹配行（通常即编译错误信息行），SIS 为 该行创建 SourceLink、并把每一个错误信息中给定的文件（和行号）作为 link 目的地，这对于我们修改源代码错误非常有帮助。

2、替换（Replace） VS 上下文敏感的智能重命名（Context-Sensitive Smart Rename）

2.1 替换（Replace）

目前来说，普通的替换命令、快捷键为 ctrl+H，足以已满足工作要求。

在弹出的替换窗口中，在 Search 框中勾选 Selection 来只在所选文本区域中替换（当然这时你要先选定区域然后再按 ctrl+H）、勾选 WholeFile 来在整个当前文件内替换、两者都不勾选来从当前光标处替换至文件末尾；点右边的 Files...按钮，可选择替换多个文件的内容。

2.2 上下文敏感的智能重命名（Context-Sensitive Smart Rename）

Smart Rename 命令、快捷键是 Ctrl+'，是上下文敏感的全局搜索替换。它可以智能地重命名全部项目文件中的一个标示符。SourceInsight 的搜索索引（search index）使得搜索过程进行地

非常快。而且，使用 Smart Rename 所做的替换会被记录在 Search Results 窗口中，每一条替换记录旁有一个 SourceLink 链接到替换发生地文件。

Smart Rename 可以用来重命名标记（symbol）。如果勾选了 Smart Reference Matching 选项，Smart Rename 就只在正确的上下文范围内进行重命名。它可以智能地重命名全部项目文件中的一个标示符；它可以重命名函数本地变量，类或结构体成员、函数。

在弹出的 Smart Rename 窗口中有下面几项：

ne 填旧名称。光标下的词会被自动加载；光标的位置非常重要，这是因为 Source Insight 会根据本地上下文背景、准确地确定你想要重命名哪一个标记。
单个词、而不是字符串。
命名成员变量、或本地变量（），Old Name 框中会显示完全标记名、即上层容器名+标记名。例如，框中的"DocDraw.paintStruc"代表 DocDraw 是函数名，paintStruc 是函数的本地成员变量。
New Name 填新名称。只填标记名，不填上层容器名。
Results 如果勾选，搜索替换结果日志会被输出到 Search Results 窗口中。可以通过 Windows 菜单来切换，或 ctrl+tab 切换察看。并且每一条记录旁会有 SourceLink 链接到替换发生地文件。
Confirm Each Replacement 每次替换询问。
Skip Comments 不重名注释部分。

【使用心得列表】

（１）如何用 Smart Rename 重命名数组的数组名？如果只选取数组名，会报错！

（２）如果勾掉 Smart Reference Matching，会搜索全部项目文件，并且 Old Name 框中不显示完全限定名；如果勾选 Smart Reference Matching，无法重命名数组名，而且鼠标位置不正确时会报错。应该如何应对？

3、在 SourceInsight 中提供的正则表达式

3.1 在 SourceInsight 中提供的正则表达式

正则表达式，是用来匹配复杂模式的特殊搜索用字符串。正则表达式串中，许多字符具有特殊的含义。例如，有个特殊的字符代表 "行首"。

下面是 SourceInsight 提供的所有可用特殊字符：

Table 4.3: Regular Expression Characters	
Character	Matches
^ (at the beginning)	beginning of line。如^Hello，匹配 Hello 在句首。
.	any single character
[abc]	any single character that belongs to the set abc
[^abc]	any single character that does not

	belong to the set abc
*	zero or more occurrences of the preceding character
+	a tab character
s	a space character
w	white space (a tab or a space character)
$	the end of the line。如 TRUE$，匹配 TRUE 在句尾。
\	转义字符。如果在它后面有元字符，取消其特殊含义。

可利用 "("和 ")"、把正则表达式分割成不同的**组**；模式中的每个组自左向右指定为 Group #n，n=1,2,…；**组的概念**在替换时很有用。

例如：

abc(xyz)可匹配 abcxyz，其中 xyz 被认为是 group#1，

利用 21 来替换(abc)(xyz)，替换结果为 xyzabc。

3.2 正则表达式在配置 tc 编译器中的应用：

正则表达式格式与源代码文件路径相对应，这里我的 tc 安装目录为 d:tc，tc 源文件放在 d:tcsrc 下，并命名为 qn.c 或 qtn.c（其中 n=1,2,…）。

观察 Tc 编译器某一次输出错误信息的格式：

Error **d:tcsrcq5.c 18**: Lvalue required in function jsSort

则我们要匹配"d:tcsrcq5.c 18"部分，进一步地，按照 SourceInsight 捕捉输出并加以解析时的要求，要以组的形式、分别匹配"d:tcsrcq5.c 18"中的文件部分和行号部分：

行号([1-9][0-9]*)

空格行号 s([1-9][0-9]*)

文件名(d:tcsrc[qQ][tT][1-9][0-9]*.[cC])

全部加起来为：

```
(d:tcsrc[qQ][tT]*[1-9][0-9]*.[cC])s([1-9][0-9]*)
```

3.3 正则表达式在配置 javac 编译器中的应用：

我的 JAVA_HOME 是 c:jdk，我的 java 源文件放于 d:javasrc 中，并命名为 qn.java 或 qtn.java（其中 n=1,2,…）。

观察 JDK 编译器某一次输出错误信息的格式：

D:javasrcQ3.java:3: ';' expected

正则表达式为：

```
([dD]:javasrc[qQ][tT]*[1-9][0-9]*.java):([1-9][0-9]*)
```

4、自定义命令

自定义命令与项目相关，在一个项目中定义的所有自定义命令属于该项目、只对该项目有效（包括快捷键等）。

自定义命令类似于命令行批处理文件。SIS 允许自定义命令在后台运行；并可以捕捉自定义命令的输出、放置于文件中、或粘贴入当前光标插入点。

分别利用上面 SIS 对输出信息的处理方式，自定义命令对集成编译器相当有用，可以捕捉编译器输出并创建 SourceLink 寻错；自定义命令对于文本过滤也相当有用，可选中待过滤区块、运行 Sort 自定义命令、粘贴回选定区块、即完成文本过滤。

请按下面步骤创建自定义命令：

Options 菜单→Custom Command

→点右边 Add 钮、填入新自定义命令名称，或下拉左边 Commands、选择命令进行修改

→Run 框、填入待执行命令行，可含有特殊元字符，见后面的元字符表

→Dir 框、执行命令行时应处的目录，如不填，以源代码文件所在目录为命令执行目录

→勾选 Output 框的 Capture Output、输出被捕捉，如果勾选 Paste Output，输出被粘贴

→勾选 Control Group 框中的 Save Files First、SIS 会在运行命令前先检查文件是否保存

→勾选 Control Group 框中的 Pause When Done、SIS 会在命令结束后暂停、方便检查

→勾选 Source Links in Output 框中的 Parse Source Links，?/p>

source insight 常用宏

转自:http://blog.csdn.net/Jupin/archive/2005/02/04/281020.aspx

说明:

该宏文件实现一些编码程中能会到的功能，如添加文件头、函数说明和宏定义等，使用时能自动添加文件名、函数名和当前日期.

使用说明:

1. Project->Open Project... 打开 Base 工程(该工程一般在"我的文档\Source Insight\Projects\Base"中);

2. Project->Add and Remove Project Files... 加入宏文件(即 Gaoke.em);

3. Options->Menu Assignments 打开 Menu Assignments 窗口，在 Command 中输入 Macro，选中要使用的宏，添加到合适的菜单中.

/*附上宏定义文件*/

/* t357.em - a small collection of useful editing macros */

/***
****
 * InsFileHeader -- insert the information of file
 *
 * modification history
 * --------------------
 * 01a, 23mar2003, added DESCRIPTION by t357
 * 01a, 05mar2003, t357 written
 * --------------------
 ****
**********************************/

/*---

 I N S E R T   H E A D E R
 Inserts a comment header block at the top of the current
function.
 This actually works font-size: 12px;"> To use this, define
an environment variable "szMyName" and set it
 to your email name. eg. set szMyName=raygr

```
----------------------------------------------------------------
--------*/
macro InsFileHeader()
{
/*##############################################
#########################
  ##########################################
#########################
  #######  Set szMyName variable to your
name    ########
  #######  for example    szMyName =
"t357"     ########
  ##########################################
#########################
  ##########################################
#########################*/
 szMyName = ""
 // Get current time
 szTime = GetSysTime(1)
 Day = szTime.Day
 Month = szTime.Month
 Year = szTime.Year
```

```
 if (Day < 10)
  szDay = "0@Day@"
 else
  szDay = Day
 szMonth = NumToName(Month)
 hBuf = GetCurrentBuf()
 szpathName = GetBufName(hBuf)
 szfileName = GetFileName(szpathName)
 nlength = StrLen(szfileName)
 szInf = Ask("Enter the information of file:")
    szDescription = Ask("Enter the description of file:")
 hbuf = GetCurrentBuf()
 // begin assembling the title string
 InsBufLine(hbuf, 0,
"/*****************************************
**********************************")
 InsBufLine(hbuf, 1, " * @szfileName@ - @szInf@")
 InsBufLine(hbuf, 2, " * ")
 InsBufLine(hbuf, 3, " * Copyright 1998-2003 Guangzhou
Gaoke Communication Technology Co.,Ltd.")
 InsBufLine(hbuf, 4, " * ")
 InsBufLine(hbuf, 5, " * DESCRIPTION: - ")
```

```
 InsBufLine(hbuf, 6, " *      @szDescription@")
 InsBufLine(hbuf, 7, " * modification history")
 InsBufLine(hbuf, 8, " * --------------------")
 InsBufLine(hbuf, 9, " * 01a,
@szDay@@szMonth@@Year@, @szMyName@ written")
 InsBufLine(hbuf, 10, " * --------------------")
 InsBufLine(hbuf, 11, "
**********************************************
**********************************/")
 // put the insertion point inside the header comment
 SetBufIns(hbuf, 1, nlength + strlen(szInf) + 8)
}
/**********************************************
**********************************
 * InsFunHeader -- insert function's information
 *
 * modification history
 * --------------------
 * 01a, 23mar2003, added DESCRIPTION by t357
 * 01a, 05mar2003, t357 written
 * --------------------
 **********************************************
```

```
**********************************/
macro InsFunHeader()
{
 // Get the owner's name from the environment variable:
szMyName.
 // If the variable doesn't exist, then the owner field is
skipped.
/*##############################################
##########################
  ##########################################
##########################
  #######  Set szMyName variable to your
name    ########
  #######  for example    szMyName =
"t357"     ########
  ##########################################
##########################
  ##########################################
##########################*/
 szMyName = ""
 // Get a handle to the current file buffer and the name
 // and location of the current symbol where the cursor is.
```

```
 hbuf = GetCurrentBuf()
 szFunc = GetCurSymbol()
 ln = GetSymbolLine(szFunc)
 // Get current time
 szTime = GetSysTime(1)
 Day = szTime.Day
 Month = szTime.Month
 Year = szTime.Year
 if (Day < 10)
  szDay = "0@Day@"
 else
  szDay = Day
 szMonth = NumToName(Month)
 szInf = Ask("Enter the information of function:")
 szDescription = Ask("Enter the description of function:")
 // begin assembling the title string
 sz =
"/*********************************************
**********************************"
 InsBufLine(hbuf, ln, sz)
 InsBufLine(hbuf, ln + 1, " * @szFunc@ - @szInf@")
 InsBufLine(hbuf, ln + 2, " * DESCRIPTION: - ")
```

```
    InsBufLine(hbuf, ln + 3, " *     @szDescription@ ")
// remove by t357.     CutWord(szDescription)
 InsBufLine(hbuf, ln + 4, " * Input: ")
 InsBufLine(hbuf, ln + 5, " * Output: ")
 InsBufLine(hbuf, ln + 6, " * Returns: ")
 InsBufLine(hbuf, ln + 7, " * ")
 InsBufLine(hbuf, ln + 8, " * modification history")
 InsBufLine(hbuf, ln + 9, " * --------------------")
 InsBufLine(hbuf, ln + 10, " * 01a,
@szDay@@szMonth@@Year@, @szMyName@ written")
 InsBufLine(hbuf, ln + 11, " * --------------------")
 InsBufLine(hbuf, ln + 12, "
**********************************************
**********************************/")
 // put the insertion point inside the header comment
 SetBufIns(hbuf, ln + 1, strlen(szFunc) + strlen(szInf) + 8)
}

/**********************************************
**********************************
 * NumToName -- change the month number to name
 *
```

```
 * modification history
 * --------------------
 * 01a, 05mar2003, t357 written
 * --------------------
 ***********************************************
**********************************/
macro NumToName(Month)
{
 if (Month == 1)
  return "jan"
 if (Month == 2)
  return "feb"
 if (Month == 3)
  return "mar"
 if (Month == 4)
  return "apr"
 if (Month == 5)
  return "may"
 if (Month == 6)
  return "jun"
 if (Month == 7)
  return "jul"
```

```
 if (Month == 8)
  return "aug"
 if (Month == 9)
  return "sep"
 if (Month == 10)
  return "oct"
 if (Month == 11)
  return "nov"
 if (Month == 12)
  return "dec"
}
/*****************************************************
*********************************
 * CutWord -- auto newline
 *
 * modification history
 * --------------------
 * 01a, 24mar2003, t357 fix some bug
 * 01a, 05mar2003, t357 written
 * --------------------
 *****************************************************
*********************************/
```

```
macro CutWord(ncurLine, szInf)
{
 LENGTH = 63
 nlength = StrLen(szInf)
 i = 0 /* loop control */
 begin = 0 /* first character's index of current line */
 pre = 0 /* preceding word's index */
 hbuf = GetCurrentBuf()
// nline = GetBufLnCur()
 while (i < nlength)
 {
/* remove by t357
  nrow = 0
  sz = ""
  while (nrow < 80)
  {
   if (nlength < 0)
    break
   sz = Cat(sz, szInf[nrow])
   nrow = nrow + 1
   nlength = nlength - 1
  }
```

```
  InsBufLine(hbuf, nline, sz)
  szInf = szInf[nrow]
 }
*/
        c = szInf[i]
        if (" " == @c@ && (i - b < LENGTH))
        {
            pre = i
        }
        else if (" " == @c@)
        {
            szOutput = ""
            k = begin /* loop control */
            while (k < pre)
            {
                szOutput = Cat(szOutput, szInf[k])
                k = k + 1
            }
            InsBufLine(hbuf, ncurLine, sz)
            ncurLine = ncurLine + 1
            begin = pre
        }
```

```
        i = i + 1
    }
    if (h != i - 1)
    {
        szOutput = ""
        k = begin /* loop control */
        while (k < pre)
        {
            szOutput = Cat(szOutput, szInf[k])
            k = k + 1
        }
        InsBufLine(hbuf, ncurLine, sz)
        ncurLine = ncurLine + 1
    }
}
/*************************************************
**********************************
 * GetFileName -- get the filename font-size:
12px;"> return name
}
/*************************************************
**********************************
```

```
 * ReturnTrueOrFalse -- Inserts "Returns True or False" at
the current line
 *
 * modification history
 * --------------------
 * 01a, 05mar2003, t357 written
 * --------------------
 ***********************************************
*********************************/
macro ReturnTrueOrFalse()
{
 szReturns = "return True if successful or False if errors."
 hbuf = GetCurrentBuf()
 ln = GetBufLnCur(hbuf)
 szCurLine = GetBufLine(hbuf, ln)
 DelBufLine(hbuf, ln)
 InsBufLine(hbuf, ln, "@szCurLine@@szReturns@")
 SetBufIns(hbuf, ln, StrLen(szReturns) + StrLen(szCurLine)
+ 3)
}
/***********************************************
*********************************
```

```
 * InsHeaderDef -- Inserts the header define in the
headerfile
 *
 * modification history
 * --------------------
 * 01a, 05mar2003, t357 written
 * --------------------
 ***********************************************
**********************************/
macro InsHeaderDef()
{
 hBuf = GetCurrentBuf()
 szpathName = GetBufName(hBuf)
 szfileName = GetFileName(szpathName)
 szfileName = toupper(szfileName)
 nlength = StrLen(szfileName)
 i = 0 /* loop control */
 szdefineName = ""
 while (i < nlength)
 {
  if (szfileName[i] == ".")
   szdefineName = Cat(szdefineName, "_")
```

```
  else
    szdefineName = Cat(szdefineName, szfileName[i])
  i = i + 1
 }
 szdefineName = Cat("_", szdefineName)
 IfdefineSz(szdefineName)
}
/*************************************************
**********************************
 * PrintDate - print date mailto:0@Day">0@Day@"
 else
  szDay = Day
 szMonth = NumToName(Month)
 hbuf = GetCurrentBuf()
 ln = GetBufLnCur(hbuf)
 szCurLine = GetBufLine(hbuf, ln)
 DelBufLine(hbuf, ln)
 InsBufLine(hbuf, ln, "@szCurLine@
@szDay@@szMonth@@Year@")
 SetBufIns(hbuf, ln, StrLen(szCurLine) + 10)
}
```

```
// Ask user for ifdef condition and wrap it around current
// selection.
// 28mar2003, modified by t357.
// 26mar2003, modified by t357.
macro InsIfdef()
{
 sz = Ask("Enter ifdef condition:")
 if (sz != "")
 {
  // IfdefSz(sz);
  hwnd = GetCurrentWnd()
     sel = GetWndSel(hwnd)
  hbuf = GetWndBuf(hwnd)
  // get line the selection (insertion point) is font-size:
12px;">  InsBufLine(hbuf, sel.lnFirst, "")
     InsBufLine(hbuf, sel.lnFirst + 1, "@ich@#ifdef @sz@")
     InsBufLine(hbuf, sel.lnFirst + 2, "@ich@" # chTab)
     InsBufLine(hbuf, sel.lnFirst + 3, "@ich@#endif  /*
@sz@ */")
     SetBufIns(hbuf, sel.lnFirst + 2, StrLen(ich) +
StrLen(chTab))
```

```
    }
}

// Wrap ifdeinef <sz> .. endif around the current selection
macro IfdefineSz(sz)
{
 hwnd = GetCurrentWnd()
 lnFirst = GetWndSelLnFirst(hwnd)
 lnLast = GetWndSelLnLast(hwnd)
 hbuf = GetCurrentBuf()
 InsBufLine(hbuf, lnFirst, "#ifndef @sz@")
 InsBufLine(hbuf, lnFirst + 1, "#define @sz@")
 InsBufLine(hbuf, lnLast + 3,  "#endif  /* @sz@ */")
 SetBufIns(hbuf, lnFirst + 2, 0)
}


/*  A U T O  E X P A N D  */
/*-------------------------------------------------------------
-----------
    Automatically expands C statements like if, for, while,
switch, etc..
```

To use this macro,

1. Add this file to your project or your Base project.
2. Run the Options->Key Assignments command and assign a
convenient keystroke to the "AutoExpand" command.
3. After typing a keyword, press the AutoExpand keystroke to have the
statement expanded. The expanded statement will contain a ### string
which represents a field where you are supposed to type more.
The ### string is also loaded in to the search pattern so you can
use "Search Forward" to select the next ### field.

For example:

1. you type "for" + AutoExpand key
2. this is inserted:

```
for (###; ###; ###)
{
 ###
}
```

3. and the first ### field is selected.

```
------------------------------------------------------------
--------*/
/************************************************
*********************************
 * AutoExpand - Automatically expands C statements
 *
 * DESCRIPTION: - Automatically expands C statements
like if, for, while,
 *    switch, etc..
 *
 * Input:
 * Output:
 * Returns:
 *
 * modification history
 * --------------------
 * 01a, 27mar2003, t357 modified
 * --------------------
 ***********************************************
*********************************/
macro AutoExpand()
{
```

```
 // get window, sel, and buffer handles
 hwnd = GetCurrentWnd()
 if (hwnd == 0)
  stop
 sel = GetWndSel(hwnd)
 if (sel.ichFirst == 0)
  stop
 hbuf = GetWndBuf(hwnd)
 // get line the selection (insertion point) is font-size:
12px;"> // parse word just to the left of the insertion point
 wordinfo = GetWordLeftOfIch(sel.ichFirst, szLine)
 ln = sel.lnFirst;
 chTab = CharFromAscii(9)
 // prepare a new indented blank line to be inserted.
 // keep white space font-size: 12px;"> szLine =
strmid(szLine, 0, ich)
 sel.lnFirst = sel.lnLast
 sel.ichFirst = wordinfo.ich
 sel.ichLim = wordinfo.ich
 // expand szWord keyword...

 if (wordinfo.szWord == "if" ||
```

```
 wordinfo.szWord == "while" ||
 wordinfo.szWord == "elseif")
 {
 SetBufSelText(hbuf, " (###)")
 InsBufLine(hbuf, ln + 1, "@szLine@" # "{");
 InsBufLine(hbuf, ln + 2, "@szLine@" # chTab);
 InsBufLine(hbuf, ln + 3, "@szLine@" # "}");
 }
else if (wordinfo.szWord == "for")
 {
 SetBufSelText(hbuf, " (###)")
 InsBufLine(hbuf, ln + 1, "@szLine@" # "{");
 InsBufLine(hbuf, ln + 2, "@szLine@" # chTab);
 InsBufLine(hbuf, ln + 3, "@szLine@" # "}");
 }
else if (wordinfo.szWord == "switch")
 {
 SetBufSelText(hbuf, " (###)")
 InsBufLine(hbuf, ln + 1, "@szLine@" # "{")
 InsBufLine(hbuf, ln + 2, "@szLine@" # "case ")
 InsBufLine(hbuf, ln + 3, "@szLine@" # chTab)
 InsBufLine(hbuf, ln + 4, "@szLine@" # chTab # "break;")
```

```
 InsBufLine(hbuf, ln + 5, "@szLine@" # "default:")
 InsBufLine(hbuf, ln + 6, "@szLine@" # chTab)
 InsBufLine(hbuf, ln + 7, "@szLine@" # "}")
 }
else if (wordinfo.szWord == "do")
 {
 InsBufLine(hbuf, ln + 1, "@szLine@" # "{")
 InsBufLine(hbuf, ln + 2, "@szLine@" # chTab);
 InsBufLine(hbuf, ln + 3, "@szLine@" # "} while ();")
 }
else if (wordinfo.szWord == "case")
 {
 SetBufSelText(hbuf, " ###")
 InsBufLine(hbuf, ln + 1, "@szLine@" # chTab)
 InsBufLine(hbuf, ln + 2, "@szLine@" # chTab # "break;")
 }
else
 stop
SetWndSel(hwnd, sel)
LoadSearchPattern("###", true, false, false);
Search_Forward
}
```

```
/*  G E T   W O R D   L E F T   O F   I C H   */
/*-------------------------------------------------------------
-----------
    Given an index to a character (ich) and a string (sz),
    return a "wordinfo" record variable that describes the
    text word just to the left of the ich.
    Output:
     wordinfo.szWord = the word string
     wordinfo.ich = the first ich of the word
     wordinfo.ichLim = the limit ich of the word
-------------------------------------------------------------
--------*/
macro GetWordLeftOfIch(ich, sz)
{
 wordinfo = "" // create a "wordinfo" structure
 chTab = CharFromAscii(9)
 // scan backwords over white space, if any
 ich = ich - 1;
 if (ich >= 0)
  while (sz[ich] == " " || sz[ich] == chTab)
   {
```

```
  ich = ich - 1;
  if (ich < 0)
   break;
  }
// scan backwords to start of word
ichLim = ich + 1;
asciiA = AsciiFromChar("A")
asciiZ = AsciiFromChar("Z")
while (ich >= 0)
 {
 ch = toupper(sz[ich])
 asciiCh = AsciiFromChar(ch)
 if ((asciiCh < asciiA || asciiCh > asciiZ) && !IsNumber(ch))
  break // stop at first non-identifier character
 ich = ich - 1;
 }
ich = ich + 1
wordinfo.szWord = strmid(sz, ich, ichLim)
wordinfo.ich = ich
wordinfo.ichLim = ichLim;
return wordinfo
}
```