

Oracle® Solaris Studio 12.2 IDE 快速入门教程

2010年9月

- 第 2 页中的“创建项目”
- 第 8 页中的“运行项目”
- 第 9 页中的“从现有代码创建项目”
- 第 10 页中的“远程开发”
- 第 11 页中的“打包应用程序”
- 第 13 页中的“编辑源文件”
- 第 20 页中的“导航源文件”
- 第 26 页中的“创建断点”
- 第 28 页中的“调试项目”
- 第 30 页中的“在机器指令级调试”
- 第 32 页中的“通过附加到某个正在运行的程序对其进行调试”
- 第 33 页中的“调试核心转储文件”

创建项目

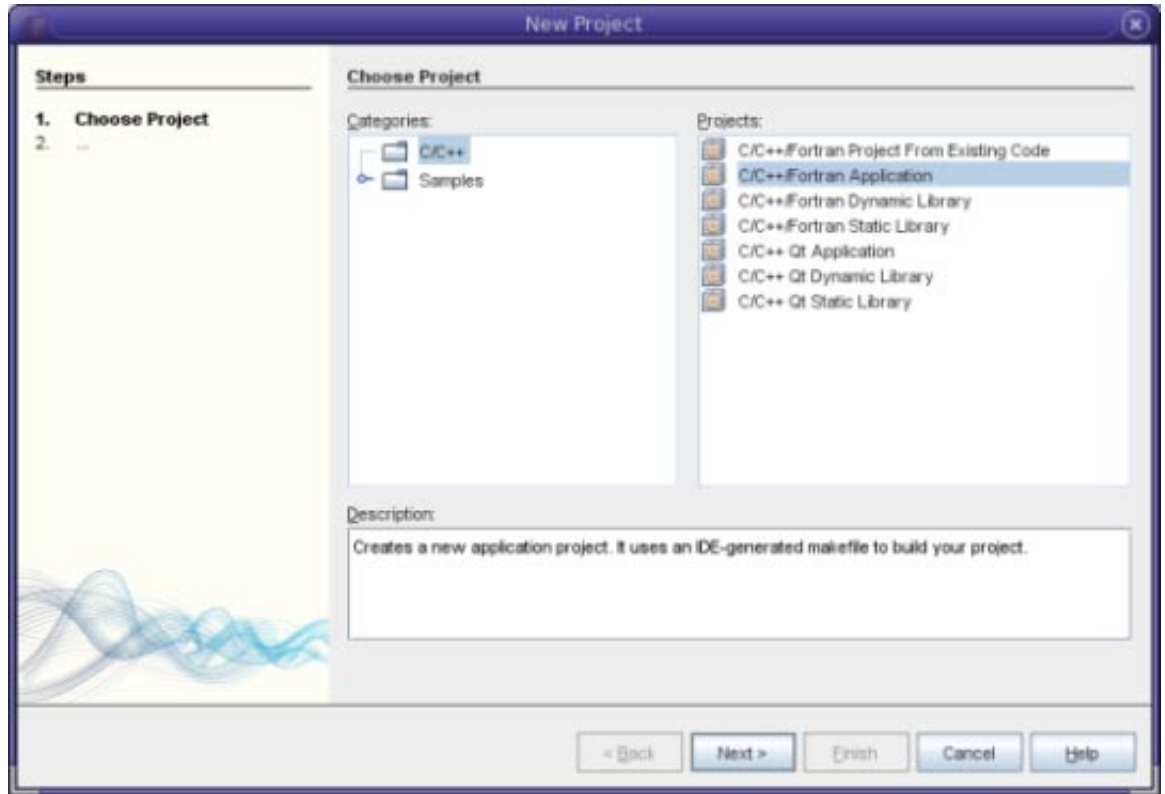
借助 Oracle Solaris Studio，您可以使用生成的 makefile 创建 C、C++ 和 Fortran 应用程序和库项目，也可以创建包含现有源代码的项目。

您可以在本地主机（从中启动 IDE 的系统）上生成、运行和调试项目，也可以在运行 Solaris 操作系统或 Linux 操作系统的远程主机上生成、运行和调试项目。

通过 C/C++/Fortran 应用程序、动态库或静态库项目，IDE 可以控制生成、运行和调试应用程序的各个方面。可在创建项目时指定项目设置，也可以在“Project Properties”（项目属性）对话框中指定。IDE 会生成一个 makefile，您的所有设置都存储在其中。

创建应用程序项目

1. 通过选择“File”（文件）>“New Project”（新建项目），打开“New Project”（新建项目）向导。
2. 在向导中选择“C/C++”类别。
3. 该向导提供了多种项目类型。选择“C/C++/Fortran Application”（C/C++/Fortran 应用程序），然后单击“Next”（下一步）。



4. 通过向导使用缺省值创建 C/C++/Fortran 应用程序新项目。您可以选择项目的名称和位置。
5. 单击 "Finish"（完成）退出向导。

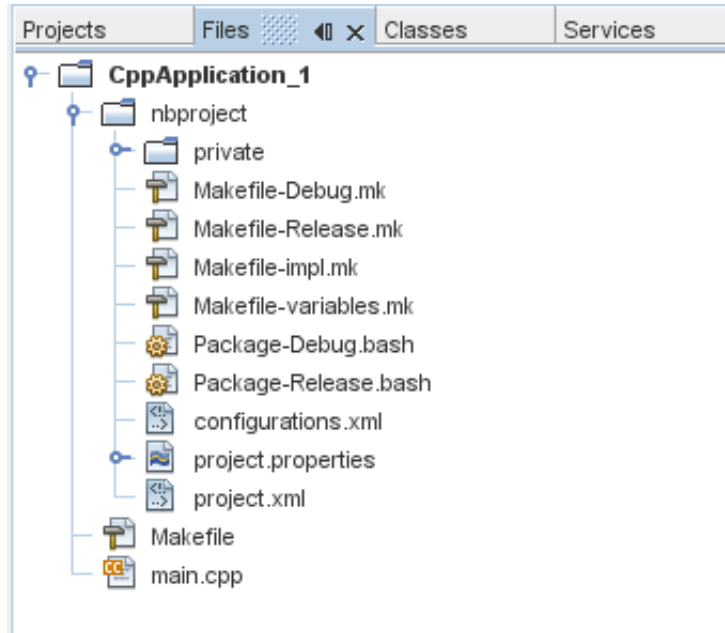
现在便创建了一个包含几个逻辑文件夹的项目。逻辑文件夹不是目录，而是一种文件组织方式，不反映文件在磁盘上的物理存储位置。添加到逻辑文件夹的文件会自动成为项目的组成部分，并在您生成项目时进行编译。

添加到 "Important Files" 文件夹的文件不是项目的组成部分，在您生成项目时不会进行编译。这些文件仅供参考，如果您有一个包含现有 makefile 的项目，这些文件将非常方便。

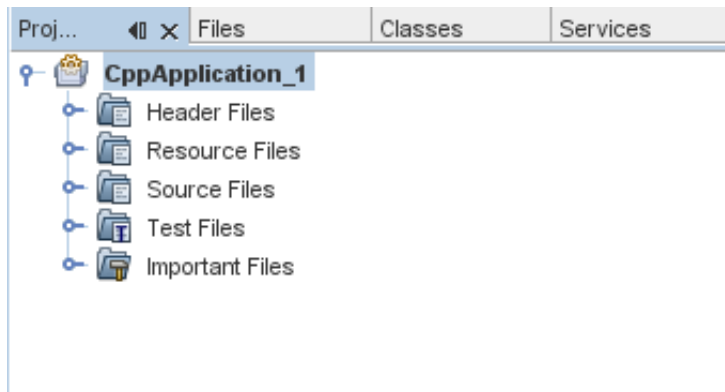
在项目的逻辑视图和物理视图之间切换

项目既有逻辑视图，也有物理视图。您可以在项目的逻辑视图和物理视图之间切换。

1. 选择 "Files"（文件）选项卡。此窗口显示项目的物理视图。该视图显示文件和文件夹，就像存储在磁盘上一样。



2. 选择 "Projects" (项目) 选项卡。此窗口显示项目的逻辑视图。



向项目添加文件和文件夹

可以向项目添加逻辑文件夹。

1. 右键单击 CppApplication_1 项目的项目节点，然后选择 "New Logical Folder" (新建逻辑文件夹)。项目中将添加一个新逻辑文件夹。
2. 右键单击新的逻辑文件夹，然后选择 "Rename" (重命名)。键入您希望新文件夹使用的名称。

可以向现有文件夹添加文件和文件夹。逻辑文件夹可以嵌套。

向项目添加新文件

可以向项目添加新文件。

1. 右键单击 "Source Files" 文件夹，然后选择 "New" (新建) > "Main C File" (主 C 文件)。
2. 在 "Name and Location" (名称和位置) 页面上，"File Name" (文件名) 字段中显示 newfile。
3. 单击 "Finish" (完成)。

此时在磁盘上的项目目录中创建了 newfile.c 文件，并添加到 "Source Files" 文件夹中。可以向此文件夹添加任何种类的文件，而不仅仅是源文件。

向项目添加更多新文件

1. 右键单击 "Header Files" 文件夹，然后选择 "New" (新建) > "C Header File" (C 头文件)。
2. 在 "Name and Location" (名称和位置) 页面上，"File Name" (文件名) 字段中显示 `newfile`。
3. 单击 "Finish" (完成)。

此时在磁盘上的项目目录中创建了 `newfile.h` 文件，并添加到 "Header Files" 文件夹中。

向项目添加现有文件

可以使用两种方法向项目添加现有文件：

- 右键单击 "Source Files" 文件夹，然后选择 "Add Existing Item" (添加现有项)。可以使用 "Select Item" (选择项) 对话框指向磁盘上的某个现有文件，然后将该文件添加到项目中。
- 右键单击 "Source Files" 文件夹，然后选择 "Add Existing Items from Folders" (从文件夹添加现有项)。使用 "Add Folders" (添加文件夹) 对话框添加包含现有文件的文件夹。

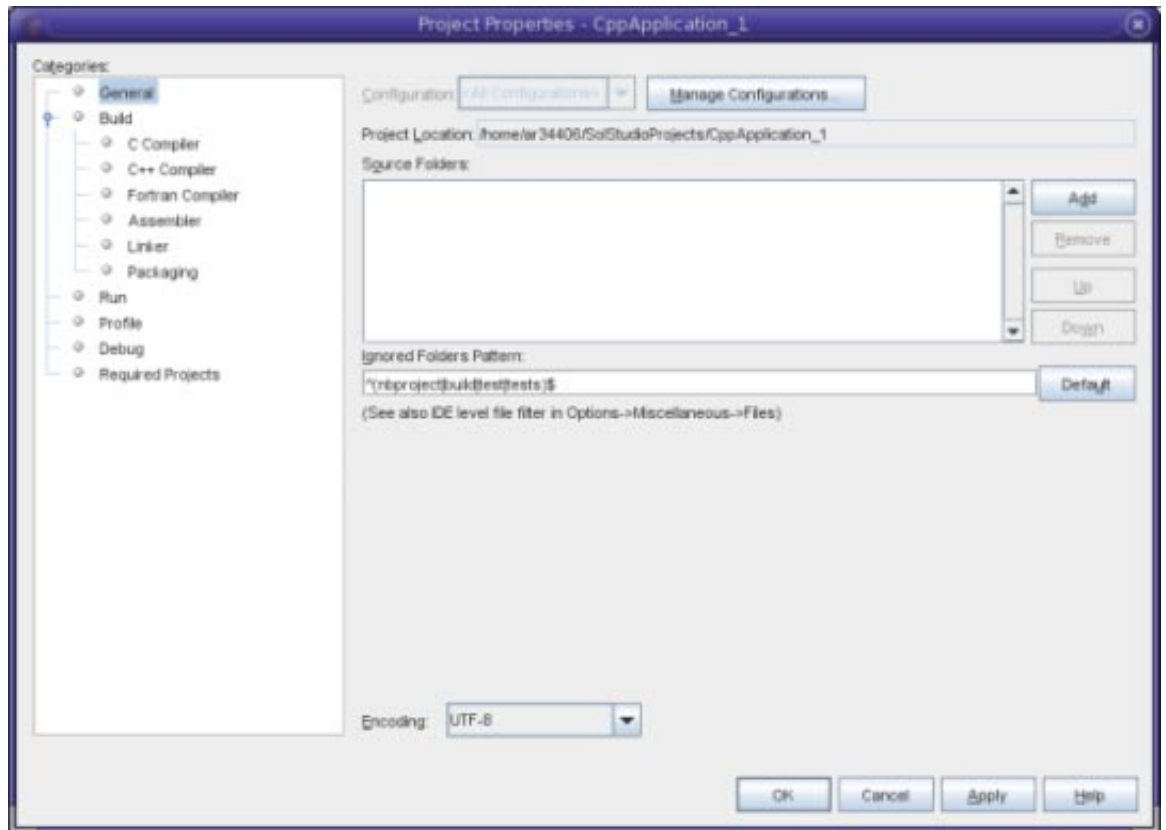
请不要使用 "New" (新建) 菜单项来添加现有项。"Name and Location" (名称和位置) 面板将显示该文件已存在。

设置项目属性

在创建项目时，有两个配置："Debug" (调试) 和 "Release" (发行)。配置是用于项目的设置集合，使您能够轻松地在许多属性设置之间快速切换。"Debug" (调试) 配置生成一个包含调试信息的应用程序版本。"Release" (发行) 配置生成一个优化版本。

"Project Properties" (项目属性) 对话框包含项目的生成信息和配置信息。打开 "Project Properties" (项目属性) 对话框：

- 右键单击 "Application" (应用程序) 项目的项目节点，然后选择 "Properties" (属性)。



通过在 "Project Properties" (项目属性) 对话框的左面板中选择一个节点然后在右面板中修改属性, 可以修改编译器设置和其他配置设置。选择其中一些节点和属性值, 请注意可以设置的属性。设置 "General" (常规) 属性时, 这些属性将应用到项目的所有配置。设置 "Build" (生成)、"Run" (运行) 或 "Debug" (调试) 属性时, 这些属性仅应用到当前选定的配置。

管理配置

在 "Project Properties" (项目属性) 对话框中更改过的属性存储在当前配置的 makefile 中。您可以编辑缺省配置或创建新的配置。创建新配置:

1. 在 "Project Properties" (项目属性) 对话框中, 单击 "Manage Configurations" (管理配置) 按钮。
2. 在 "Configurations" (配置) 对话框中, 选择与所需配置最为匹配的配置。这种情况下, 请选择 "Release" 配置, 并单击 "Copy" (复制) 按钮, 然后单击 "Rename" (重命名)。
3. 在 "Rename" (重命名) 对话框中, 将配置重命名为 "PerformanceRelease"。单击 "OK" (确定)。
4. 在 "Configurations" (配置) 对话框中, 单击 "OK" (确定)。
5. 在 "Project Properties" (项目属性) 对话框的左面板中选择 "C Compiler" (C 编译器) 节点。请注意, 在 "Configuration" (配置) 下拉式列表中已选中 "PerformanceRelease" 配置。
6. 在右面板的属性表中, 将 "Development Mode" (开发模式) 由 "Release" 更改为 "PerformanceRelease"。单击 "OK" (确定)。

您现已创建一个新配置, 该配置将使用一组不同的选项来编译应用程序。

设置源文件属性

设置项目的项目属性时, 相关属性将应用到项目中的所有文件。可以为某个特定文件设置一些属性。

1. 右键单击 newfile.c 源文件, 然后选择 "Properties" (属性)。
2. 在 "Categories" (类别) 面板中单击 "General" (常规) 节点, 您将可以选择一个不同的编译器或其他工具来生成此文件。还可以选择一个复选框, 以便从当前选定项目配置的生成中排除该文件。

- 单击 "C Compiler" (C 编译器) 节点, 您将可以覆盖项目编译器设置以及此文件的其他属性。
- 取消 "Project Properties" (项目属性) 对话框。

设置主项目

在 "Projects" (项目) 窗口中右键单击某个项目节点时, 将显示一个弹出菜单, 其中包含您可以对选定项目执行的操作。如果同时打开了多个项目, 对某个项目节点显示弹出菜单表明您正在针对该项目执行操作。

菜单栏和工具栏上大多数与项目相关的操作都针对主项目。在 "Projects" (项目) 窗口中, 主项目节点以粗体显示。

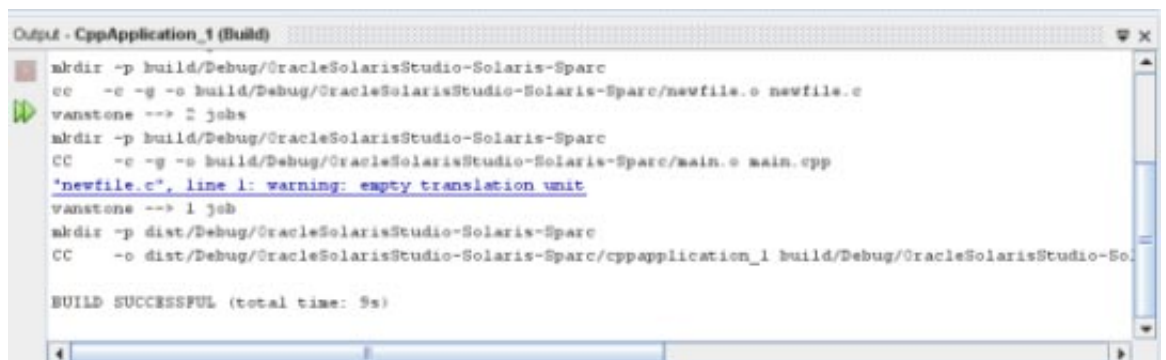
在 IDE 中更改主项目:

- 右键单击所需的项目节点, 然后选择 "Set as Main Project" (设置为主项目)。此项目现在是 IDE 中的主项目, 菜单栏和工具栏中的操作都针对此项目。

生成项目

生成项目:

- 右键单击项目, 然后选择 "Build" (生成), 即可生成项目。生成输出显示在 "Output" (输出) 窗口中。



```
Output - CppApplication_1 (Build)
mkdir -p build/Debug/OracleSolarisStudio-Solaris-Sparc
cc -c -g -o build/Debug/OracleSolarisStudio-Solaris-Sparc/newfile.o newfile.c
vanstone --> 2 jobs
mkdir -p build/Debug/OracleSolarisStudio-Solaris-Sparc
CC -c -g -o build/Debug/OracleSolarisStudio-Solaris-Sparc/main.o main.cpp
'newfile.c', line 1: warning: empty translation unit
vanstone --> 1 job
mkdir -p dist/Debug/OracleSolarisStudio-Solaris-Sparc
CC -o dist/Debug/OracleSolarisStudio-Solaris-Sparc/cppapplication_1 build/Debug/OracleSolarisStudio-So.

BUILD SUCCESSFUL (total time: 9s)
```

- 在主工具栏中的配置下拉式列表中, 将配置从 "Debug" (调试) 切换到 "PerformanceRelease"。现在将使用 "PerformanceRelease" 配置生成项目。
- 右键单击项目, 然后选择 "Build" (生成), 即可生成项目。生成输出显示在 "Output" (输出) 窗口中。

要同时生成项目的多个配置, 请选择 "Run" (运行) > "Batch Build Main Project" (批量生成主项目), 然后在 "Batch Build" (批量生成) 对话框中选择要生成的配置。

通过右键单击项目并从菜单中选择操作, 可以生成项目、清除项目或者清除并生成项目。项目还会将不同配置中的对象文件和可执行文件分开, 这样您就不必担心来自多个配置的文件混在一起。

编译单一文件

编译单一源文件:

- 右键单击 newfile.c 文件并选择 "Compile File" (编译文件), 将仅编译此文件。

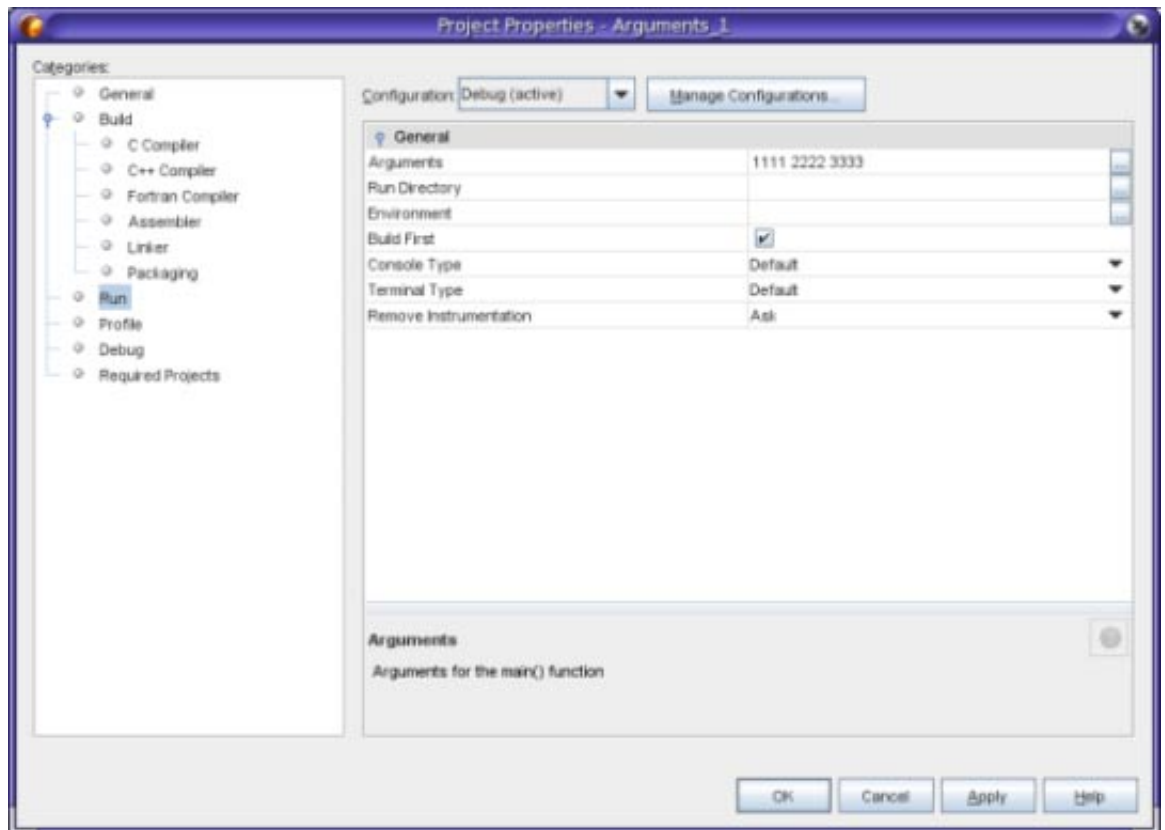
注 - 项目类型 "C/C++/Fortran Project From Existing Code" (源自现有代码的 C/C++/Fortran 项目) 不支持单一文件编译。

运行项目

Arguments 样例程序会输出命令行参数。在运行程序之前，您需要在当前配置中设置一些参数，然后运行程序。

要创建 Arguments_1 项目，请设置一些参数，然后运行该项目：

1. 选择 "File" (文件) > "New Project" (新建项目)。
2. 在项目向导中，展开 "Samples" (样例) 类别。
3. 选择 "C/C++" 子类别，然后选择 Arguments 项目。单击 "Next" (下一步)，然后单击 "Finish" (完成)。
4. 右键单击 Arguments_1 项目节点，然后选择 "Build" (生成)，即可生成项目。
5. 右键单击 Arguments_1 项目节点，然后选择 "Properties" (属性)。
6. 在 "Project Properties" (项目属性) 对话框中，选择 "Run" (运行) 节点。
7. 在 "Arguments" (参数) 文本字段中，键入 1111 2222 3333。单击 "OK" (确定)。



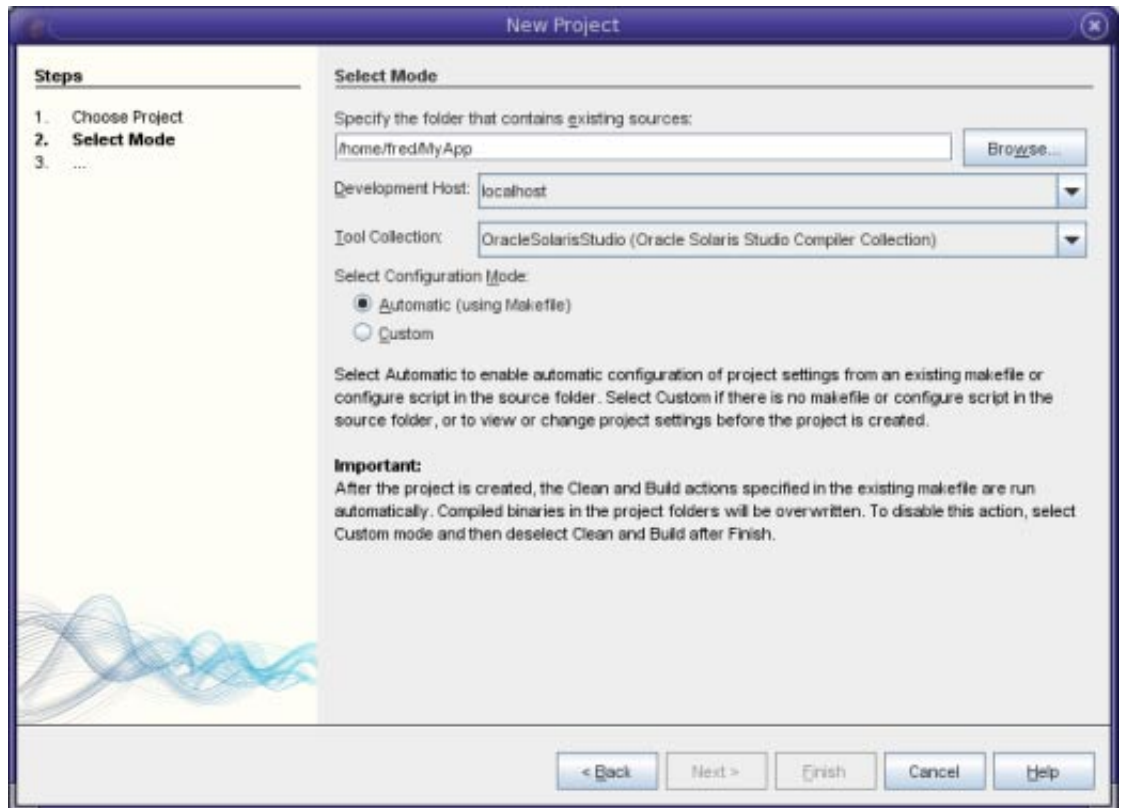
8. 选择 "Run" (运行) > "Run Main Project" (运行主项目)。应用程序将生成并运行。您的参数显示在一个外部窗口中。

提示 - 当您运行项目时，"Run Monitor" (运行监视器) 选项卡将打开，并显示用于观察应用程序行为的配置工具。您可以在 "Project Properties" (项目属性) 对话框中的 "Profile" (配置) 类别中关闭这些配置工具。

从现有代码创建项目

借助 "C/C++/Fortran Project From Existing Code" (源自现有代码的 C/C++/Fortran 项目)，IDE 可依靠现有的 makefile 获取有关如何编译和运行应用程序的说明。

1. 选择 "File" (文件) > "New Project" (新建项目)。
2. 选择 "C/C++" 类别。
3. 选择 "C/C++/Fortran Project From Existing Code" (源自现有代码的 C/C++/Fortran 项目)，然后单击 "Next" (下一步)。
4. 在 "New Project" (新建项目) 向导的 "Select mode" (选择模式) 页面上，单击 "Browse" (浏览) 按钮。在 "Select Project Folder" (选择项目文件夹) 对话框中，导航到源代码所在的目录，单击 "Select" (选择)。



5. 使用缺省配置模式 "Automatic" (自动)。单击 "Finish" (完成)。
6. 项目将在 "Projects" (项目) 窗口中创建并打开，然后 IDE 会自动运行在现有 makefile 中指定的 "Clean" (清除) 和 "Build" (生成) 部分。项目还会自动配置以获取代码帮助。

项目将在 "Projects" (项目) 窗口中创建并打开。您现已创建一个项目，是包装现有代码的一个瘦包装器。

生成和重新生成项目

生成项目：

- 右键单击项目的项目节点，然后选择 "Build" (生成)。

重新生成项目：

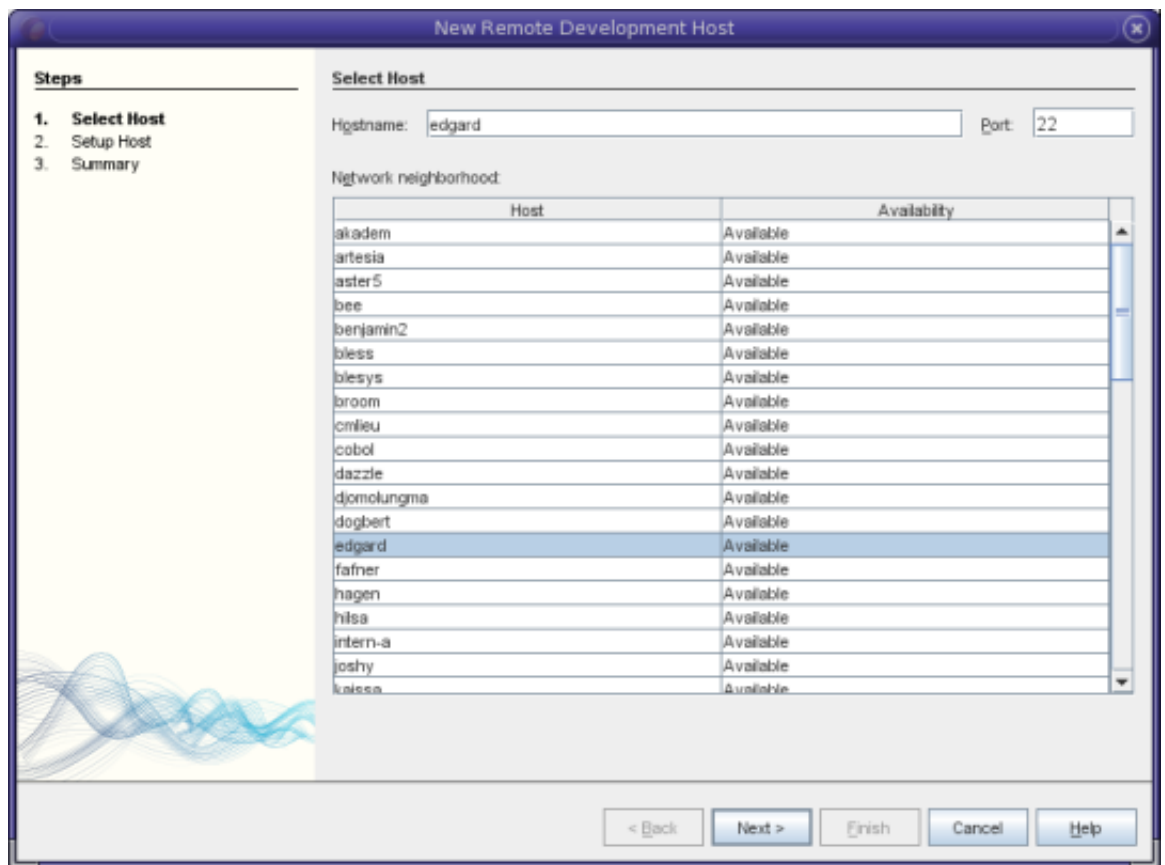
- 右键单击项目的项目节点，然后选择 "Clean and Build"（清除并生成）。

远程开发

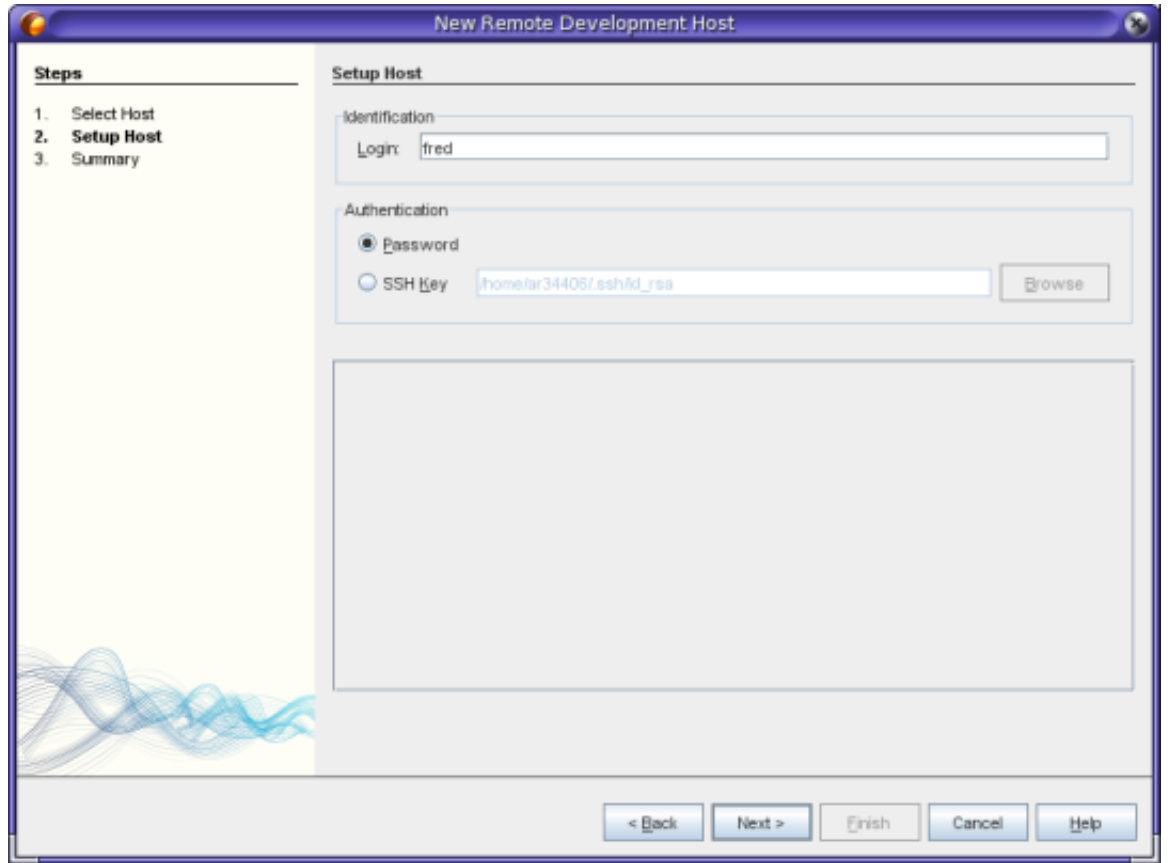
您可以在本地主机（从中启动 IDE 的系统）上生成、运行和调试项目，也可以在运行 UNIX® 操作系统的远程主机上生成、运行和调试项目。

您可以在 "Options"（选项）对话框的 "Build Tools"（生成工具）选项卡中定义远程开发主机。添加远程主机：

1. 选择 "Tools"（工具）> "Options"（选项）。
2. 在 "Options"（选项）对话框的 "Build Tools"（生成工具）选项卡中，单击 "Edit"（编辑）。
3. 在 "Development Host Manager"（开发主机管理器）对话框中，单击 "Add"（添加）。
4. 在 "New Remote Development Host"（新建远程开发主机）向导的 "Select Host"（选择主机）页面上，在 "Hostname"（主机名）字段中键入主机的系统名称，或者双击 "Network neighborhood"（网上邻居）列表中的某个可用主机以将其选中。单击 "Next"（下一步）。



5. 在 "Setup Host"（设置主机）页面上，在 "Login"（登录）字段中键入登录名称，然后单击 "Next"（下一步）。



6. 向导将提示您输入口令，连接到主机，并显示 "Summary"（摘要）页面。单击 "Finish"（完成）。
7. 在 "Development Hosts Manage"（开发主机管理）对话框中将主机添加到 "Development Hosts"（开发主机）列表中，然后单击 "OK"（确定）。

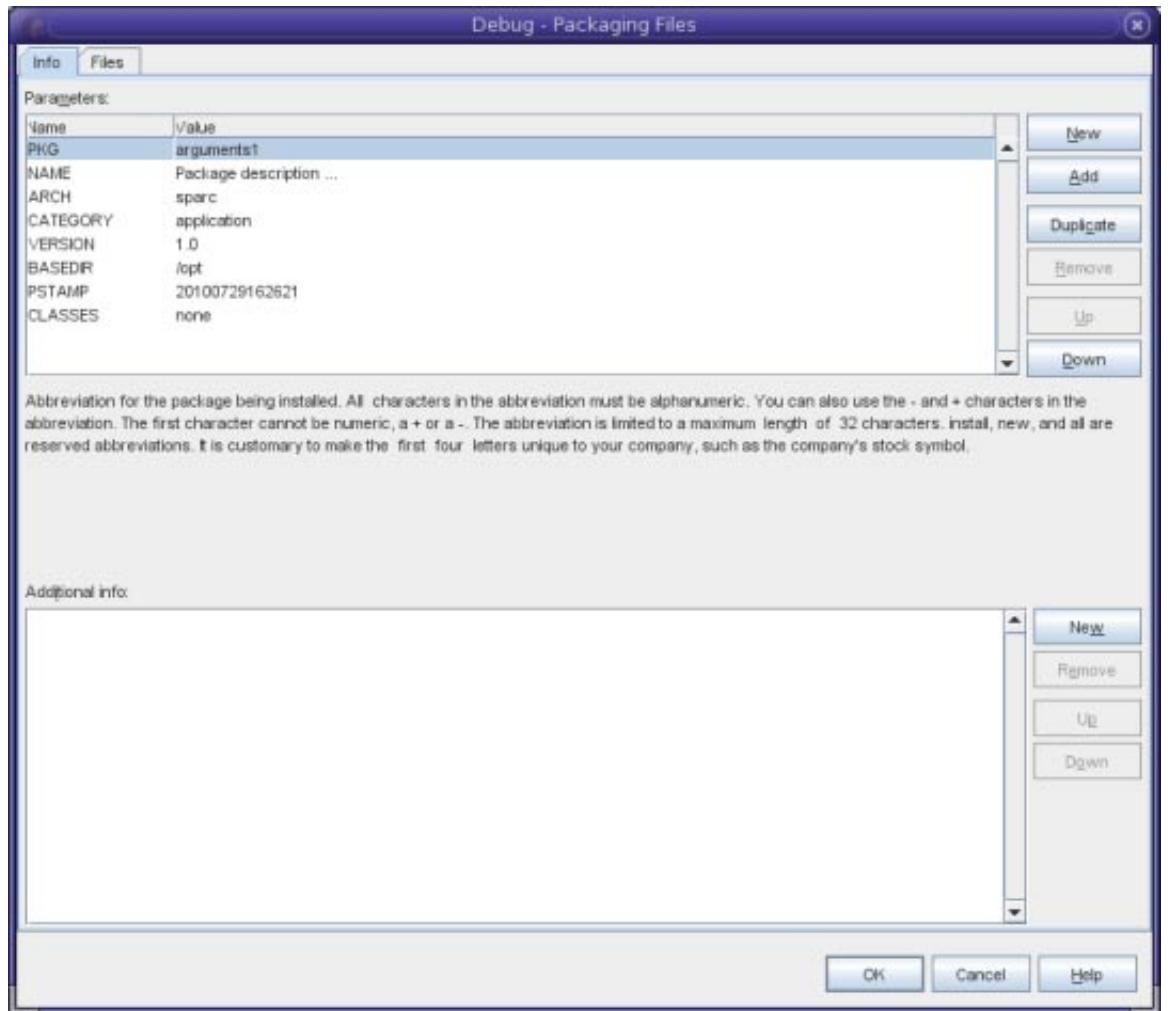
要在远程主机上开发项目，该项目必须位于在本地主机和远程主机上都可以看到的共享文件系统上。通常，这样的文件系统是使用 NFS 或 Samba 进行共享的。定义远程主机时，您可以在项目源文件的本地路径和远程路径之间定义映射。

创建项目时，选择当前开发主机作为项目的开发主机。可以在 "Project Properties"（项目属性）对话框的 "Build"（生成）面板上更改项目的开发主机。还可以在调试可执行文件或核心转储文件时指定开发主机。

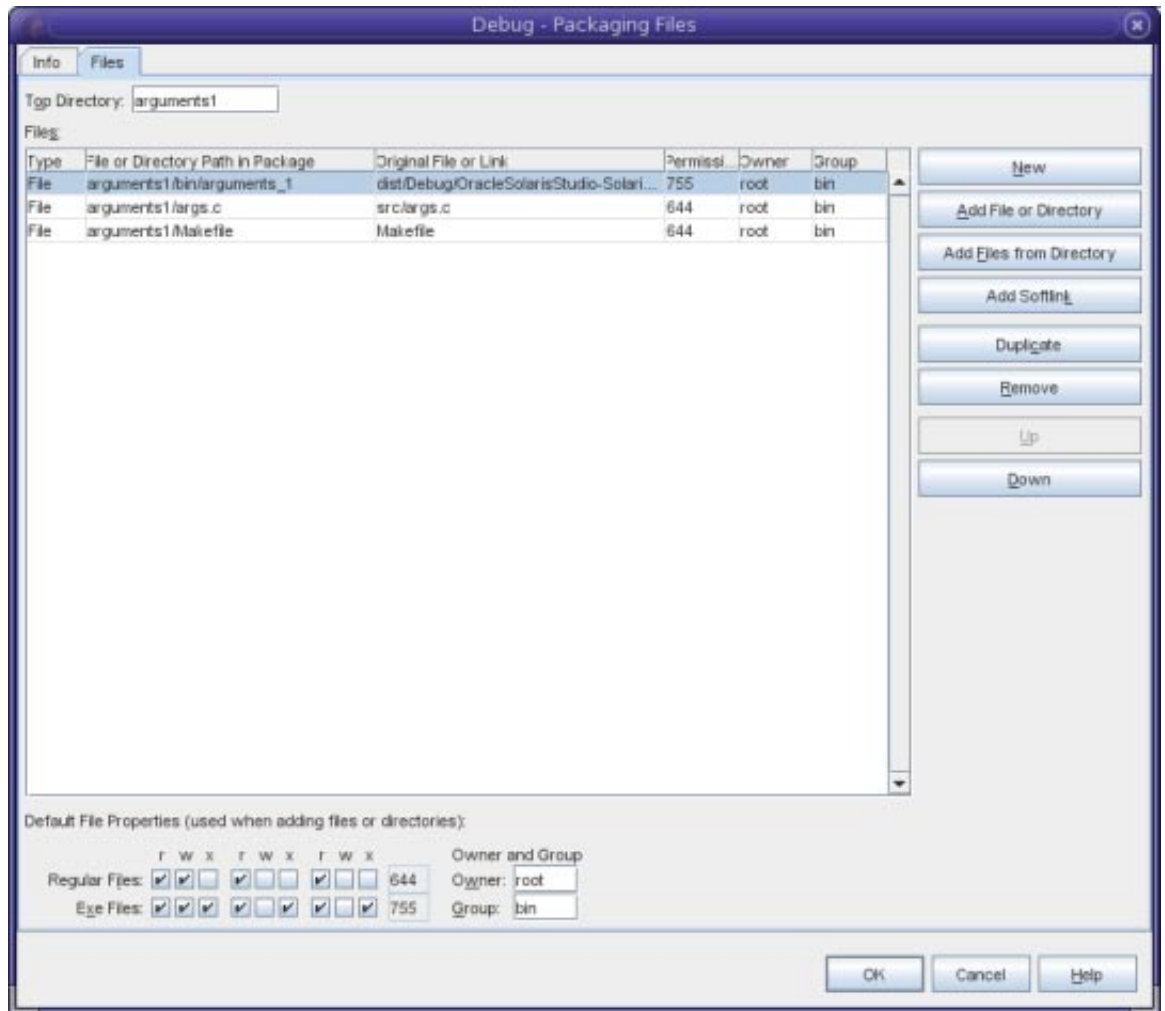
打包应用程序

可以将完成的应用程序打包为 tar 文件、zip 文件、Solaris SVR4 软件包、RPM 或 Debian 软件包。

1. 右键单击 Arguments_1 项目，然后选择 "Properties"（属性）。
2. 在 "Project Properties"（项目属性）对话框中，选择 "Packaging"（打包）节点。
3. 从下拉式列表中选择 "Solaris SVR4" 软件包类型。
4. 如果想要为软件包使用其他目标目录或文件名，请更改输出路径。
5. 单击 "Packaging Files Browse"（打包文件浏览）按钮。在 "Packaging Files"（打包文件）对话框中（对于 SVR4 软件包），根据需要在 "Info"（信息）选项卡上修改软件包参数。



6. 对于所有软件包类型，请使用 "Files"（文件）选项卡上的按钮将文件添加到软件包中。对于每个文件，可以在 "Files"（文件）列表的 "File or Directory Path in Package"（软件包中的文件或目录路径）列中指定希望包含在软件包中的路径。"Files"（文件）列表完成后，单击 "OK"（确定）。



7. 如果希望通过单击复选框来完成，请关闭详细模式。
8. 单击 "OK" (确定)。
9. 要生成软件包，请右键单击项目，然后选择 "More Build Commands" (更多生成命令) > "Build Package" (生成软件包)。

编辑源文件

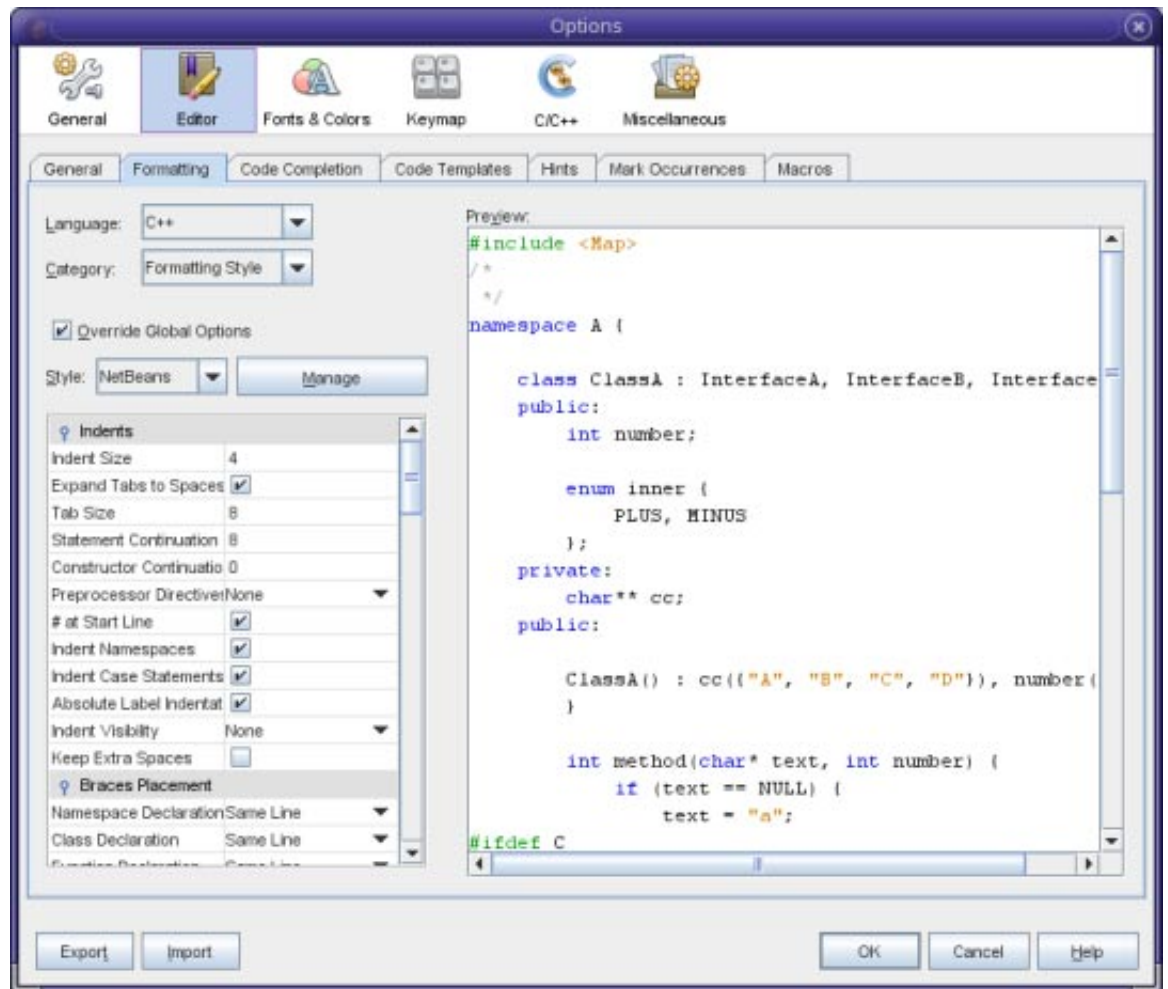
Oracle Solaris Studio IDE 提供了高级编辑和代码帮助功能，可帮助您查看和修改源代码。要了解这些功能，请使用 Quote 项目：

1. 选择 "File" (文件) > "New Project" (新建项目)。
2. 在项目向导中，展开 "Samples" (样例) 类别和 "C/C++" 子类别，然后选择 Quote 项目。单击 "Next" (下一步)，然后单击 "Finish" (完成)。

设置格式化样式

可以使用 "Options" (选项) 对话框为项目配置缺省格式化样式。

1. 选择 "Tools" (工具) > "Options" (选项)。
2. 在对话框的顶部窗格中，单击 "Editor" (编辑器)。
3. 单击 "Formatting" (格式化) 选项卡。
4. 从 "Language" (语言) 下拉式列表中，选择您需要设置格式化样式的语言。
5. 从 "Style" (样式) 下拉式列表中，选择您需要设置的样式。



6. 根据需要修改样式属性。

在 C 和 C++ 文件中折叠代码块

对于某些文件类型，您可以使用代码折叠功能折叠代码块，这样，只有块的第一行显示在程序编辑器中。

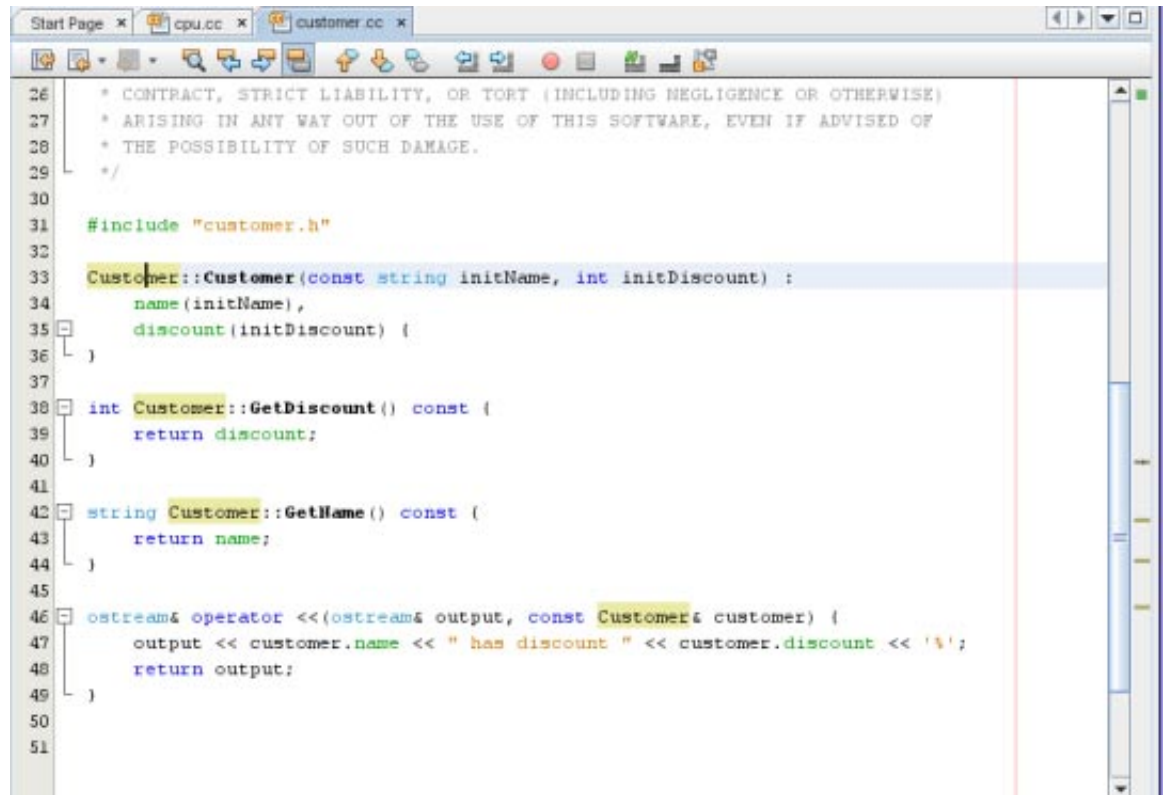
1. 在 Quote_1 应用程序项目中，打开 "Source Files" 文件夹，然后双击 cpu.cc 文件以在程序编辑器中将其打开。
2. 在左边界中单击折叠图标（带有减号的小框）可折叠其中一种方法的代码。
3. 将鼠标悬停到折叠块右侧的 {...} 符号上方可显示块中的代码。

使用语义高亮显示

您可以设置一个选项，以便当您单击某个类、函数、变量或宏时，该类、函数、变量或宏在当前文件中的每个实例都会高亮显示。

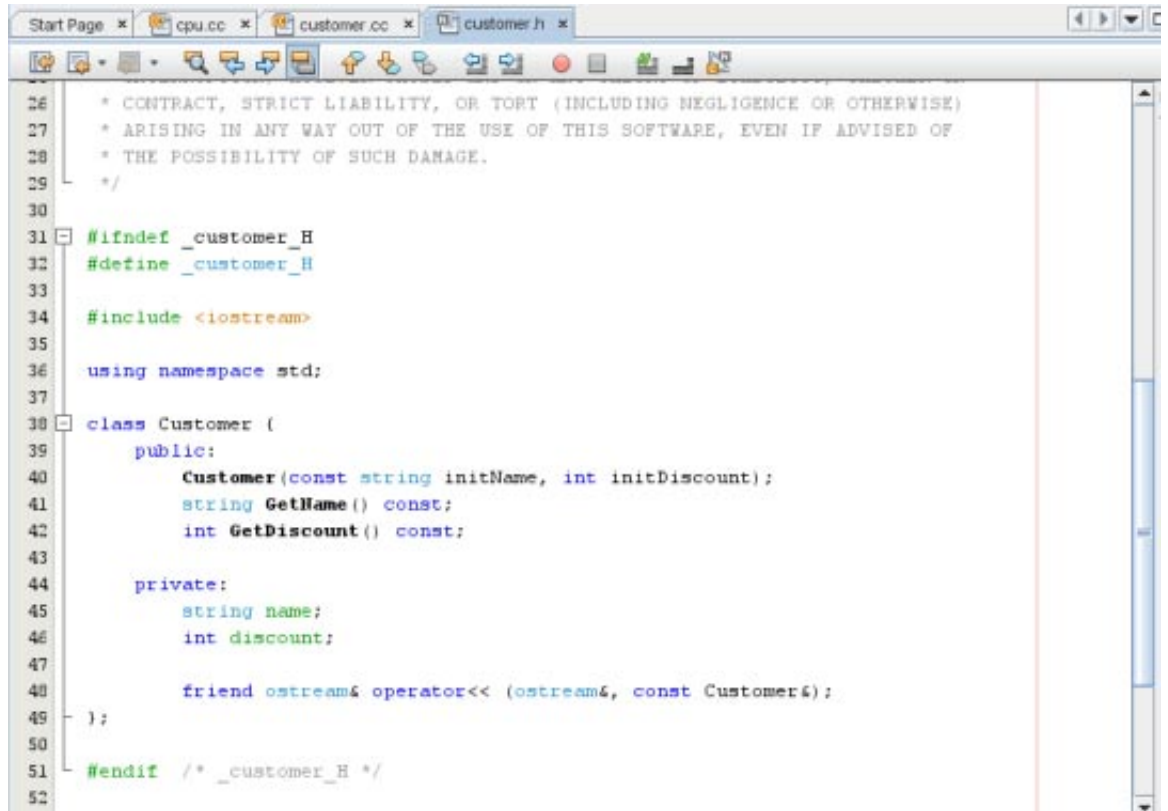
1. 选择 "Tools"（工具）> "Options"（选项）。
2. 在对话框的顶部窗格中，单击 "C/C++"。
3. 单击 "Highlighting"（高亮显示）选项卡。
4. 确保所有复选框都具有复选标记。
5. 单击 "OK"（确定）。
6. 在 Quote_1 项目的 customer.cc 文件中，请注意函数名称以粗体突出显示。

- 单击 Customer 类的某个实例。
- 文件中 Customer 类的所有实例都会以黄色背景突出显示。



```
26 | * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 | * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
28 | * THE POSSIBILITY OF SUCH DAMAGE.
29 | */
30 |
31 | #include "customer.h"
32 |
33 | Customer::Customer(const string initName, int initDiscount) :
34 |     name(initName),
35 |     discount(initDiscount) {
36 | }
37 |
38 | int Customer::GetDiscount() const {
39 |     return discount;
40 | }
41 |
42 | string Customer::GetName() const {
43 |     return name;
44 | }
45 |
46 | ostream& operator <<(ostream& output, const Customer& customer) {
47 |     output << customer.name << " has discount " << customer.discount << '\n';
48 |     return output;
49 | }
50 |
51 |
```

- 在 customer.h 文件中，请注意类字段以粗体突出显示。

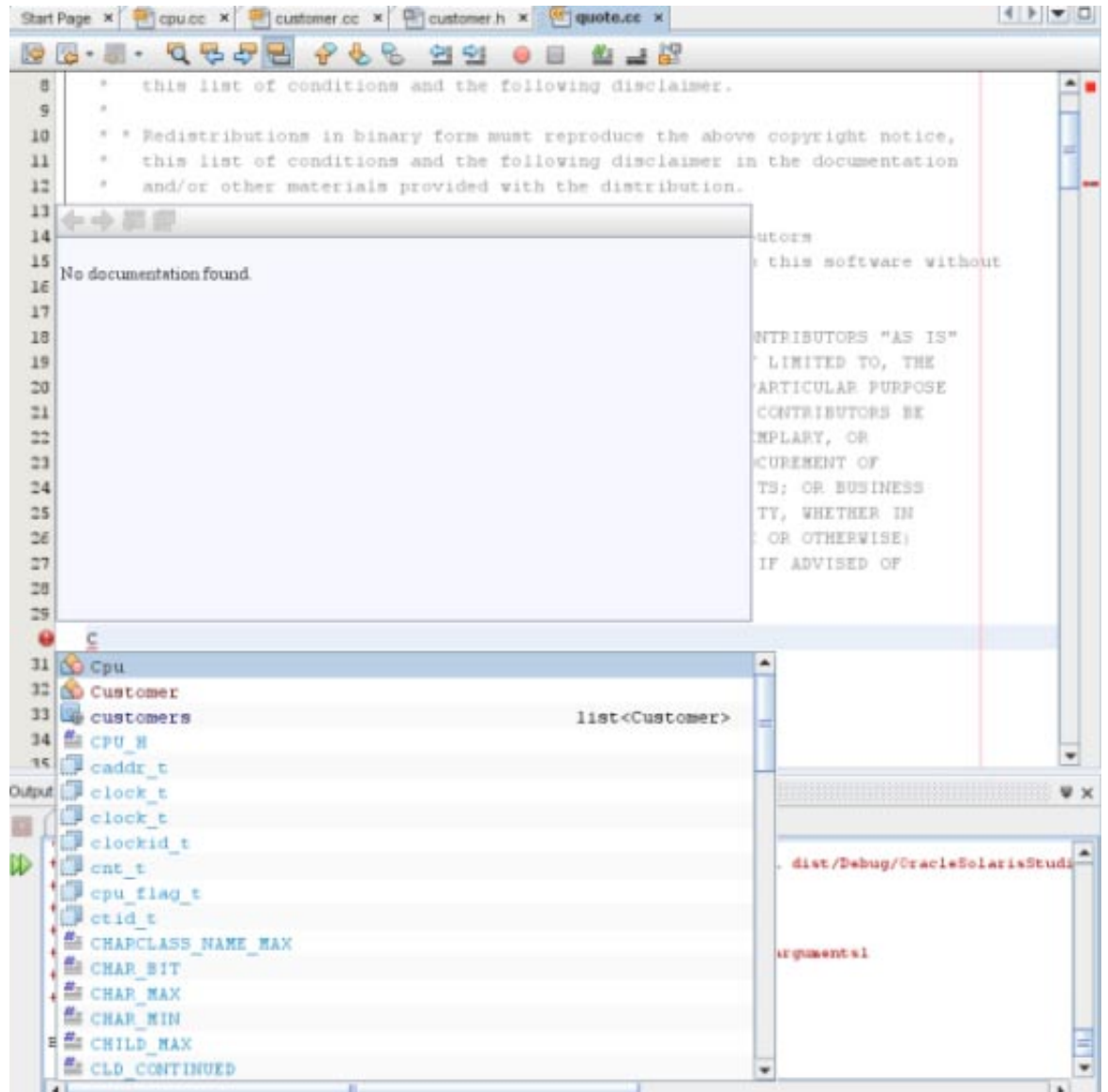


```
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
28  * THE POSSIBILITY OF SUCH DAMAGE.
29  */
30
31  #ifndef _customer_H
32  #define _customer_H
33
34  #include <iostream>
35
36  using namespace std;
37
38  class Customer {
39  public:
40      Customer(const string initName, int initDiscount);
41      string GetName() const;
42      int GetDiscount() const;
43
44  private:
45      string name;
46      int discount;
47
48      friend ostream& operator<< (ostream&, const Customer&);
49  };
50
51  #endif /* _customer_H */
52
```

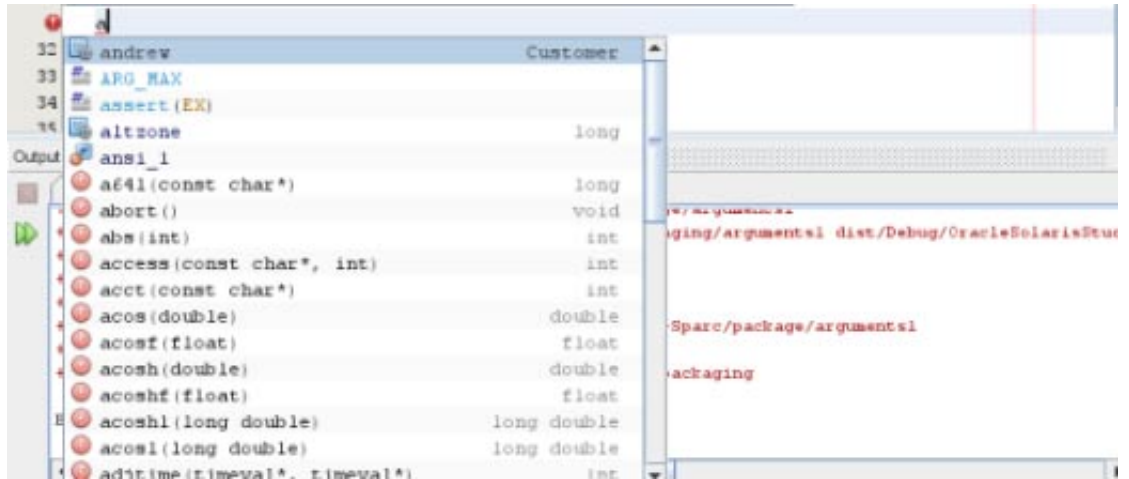
使用代码完成

IDE 有一个动态的 C 和 C++ 代码完成功能。通过该功能，您在键入一个或多个字符后，可以看到一个列表，其中列出了可用于完成表达式的可能的类、方法和变量等。

1. 打开 Quote_1 项目中的 quote.cc 文件。
2. 在 quote.cc 文件的第一个空白行上，键入大写的 C，然后按 Ctrl + 空格键。代码完成框将显示一个包含 Cpu 类和 Customer 类的简短列表。还会打开一个文档窗口，并显示消息 "No documentation found because the project source code does not include documentation"（找不到文档，因为项目源代码不包含文档）。
3. 通过再次按 Ctrl + 空格键，展开代码完成列表。



4. 从列表中选择标准库函数（如 `calloc()`），如果IDE可访问该函数的手册页，文档窗口将显示相应手册页。
5. 选择 `Customer` 类，然后按 `Enter` 键。
6. 通过键入 `andrew` 完成 `Customer` 类的新实例。在下一行上，键入字母 `a`，然后按 `Ctrl + 空格` 键。代码完成框将显示一个以字母 `a` 开头的选项列表（如方法参数、类字段和全局名称），这些选项可从当前上下文访问。



7. 双击 `andrew` 选项以接受它，然后在后面键入一个句点。系统会自动为您提供一个包含 `Customer` 类的公共方法和字段的列表。

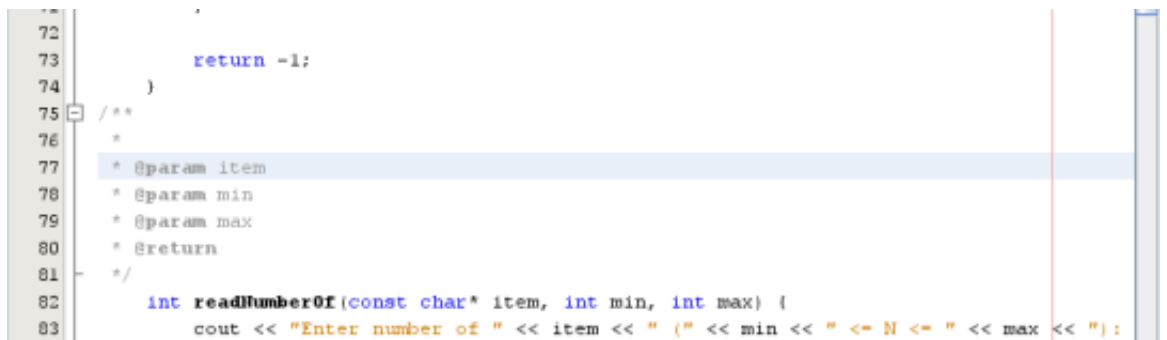


8. 删除所添加的代码。

添加源代码文档

您可以向代码添加注释，以便为函数、类和方法生成文档。IDE 可识别使用 Doxygen 语法的注释，并自动生成文档，还可以自动生成注释块来记录注释下方的函数。

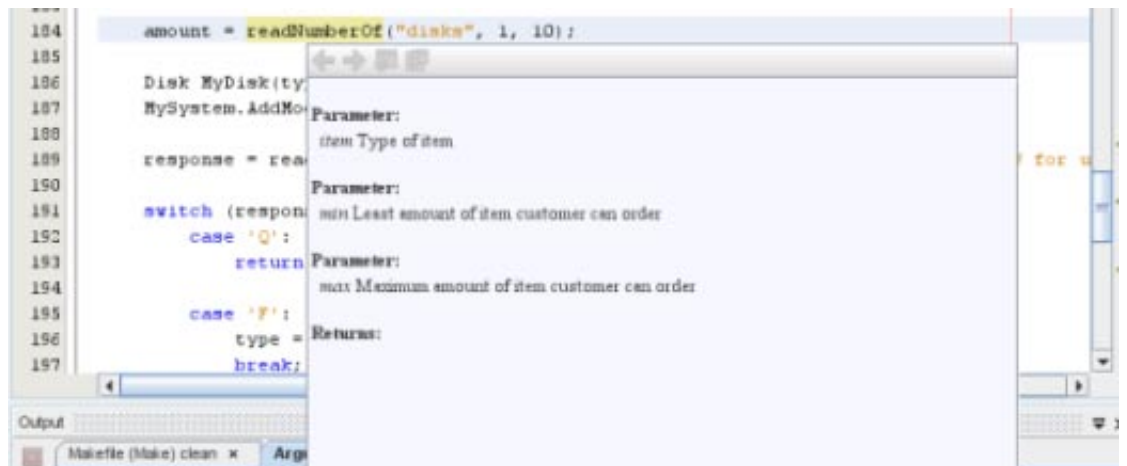
1. 在 `quote.cc` 文件中，将光标置于行 `int readNumberOf(const char* item, int min, int max) {` 上面的行上。
2. 键入一个斜杠和两个星号，然后按 `Enter` 键。编辑器将为 `readNumberOf` 类插入一个 Doxygen 格式的注释。



3. 向每个 `@param` 行添加一些描述性文本，然后保存文件。
4. 单击 `readNumberOf` 类将其以黄色突出显示，然后单击右侧的其中一个标记实例跳到使用该类的某个位置。

```
76 *
77 * @param item Type of item
78 * @param min Least amount of item customer can order
79 * @param max Maximum amount of item customer can order
80 * @return
81 */
82 int readNumberOf(const char* item, int min, int max) {
83     cout << "Enter number of " << item << " (" << min << " <= N <= " << max << "):";
84
85     string s;
```

5. 单击所跳到的行中的 readNumberOf 类，然后按 Ctrl + Shift + 空格键显示刚才为这些参数添加的文档。



6. 单击文件中的任何其他地方关闭文档窗口，然后再次单击 readNumberOf 类。
7. 选择 "Source" (源) > "Show documentation" (显示文档) 再次为该打开文档窗口。

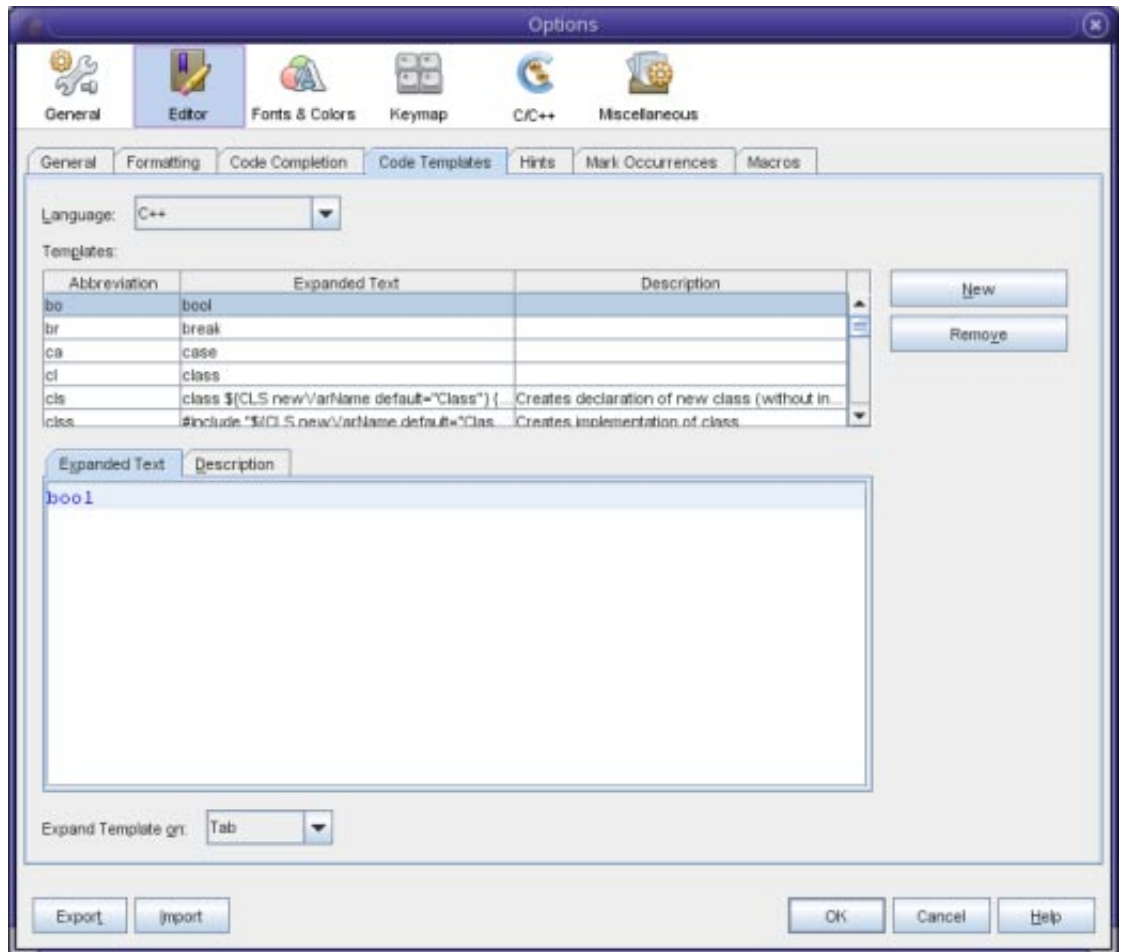
使用代码模板

程序编辑器有一组用于 C、C++ 和 Fortran 代码的常见代码片段的自定义代码模板。您可以通过键入缩写然后按 Tab 键来生成完整的代码片段。例如，在 Quote_1 项目的 quote.cc 文件中：

- 键入 uns 然后按 Tab 键，uns 会展开为 unsigned。
- 键入 iff 然后按 Tab 键，iff 会展开为 if (exp) {}。
- 键入 ifs 然后按 Tab 键，ifs 会展开为 if (exp) {} else {}。
- 键入 fori 然后按 Tab 键，fori 会展开为 for (int i=0; i< size; i++) { Object size = array[i]; }。

要看到所有可用的代码模板，修改这些模板，创建自己的代码模板，或选择其他键来展开模板，请执行下列操作：

1. 选择 "Tools" (工具) > "Options" (选项)。
2. 在 "Options" (选项) 对话框中，选择 "C/C++"，然后单击 "Code Templates" (代码模板) 选项卡。
3. 从 "Language" (语言) 下拉式列表中选择一种语言。



使用配对完成

当您编辑 C 和 C++ 源文件时，程序编辑器会对成对字符（如括号、圆括号和引号）进行智能匹配。当您键入这些字符中的其中一个时，程序编辑器会自动插入封闭字符。

1. 在 Quote_1 项目中，将光标放到 module.cc 文件的第 116 行上的 { 后面，然后按 Return 键打开一个新行。
2. 键入 enum state {，然后按 Return 键，将自动添加封闭花括号和分号，且光标位于两个花括号之间。
3. 键入 invalid=0, success=1 来完成枚举。
4. 在该行上枚举的封闭 }; 后面，键入 if (，将自动添加封闭圆括号，且光标位于两个圆括号之间。
5. 键入 v==null。在右圆括号后键入 i 并换行，将自动添加封闭括号。
6. 删除所添加的代码。

导航源文件

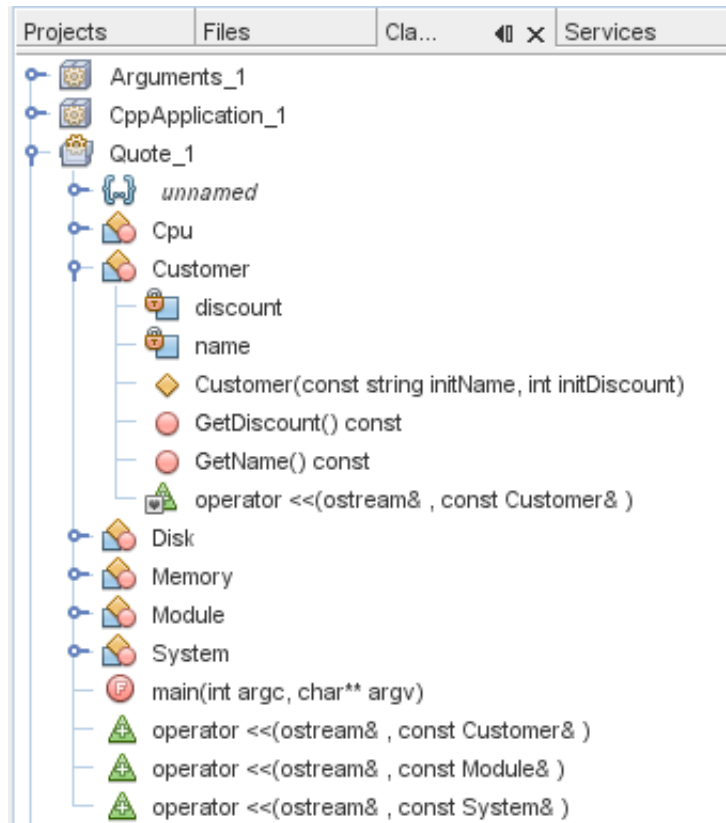
IDE 提供了用于查看源代码的高级导航功能。要了解这些功能，请继续使用 Quote_1 项目。

使用 "Classes" (类) 窗口

使用 "Classes" (类) 窗口可看到项目中的所有类以及每个类的成员和字段。

1. 单击 "Classes" (类) 选项卡可显示 "Classes" (类) 窗口。
2. 展开 Quote_1 节点。会列出项目中的所有类。

3. 展开 Customer 类。

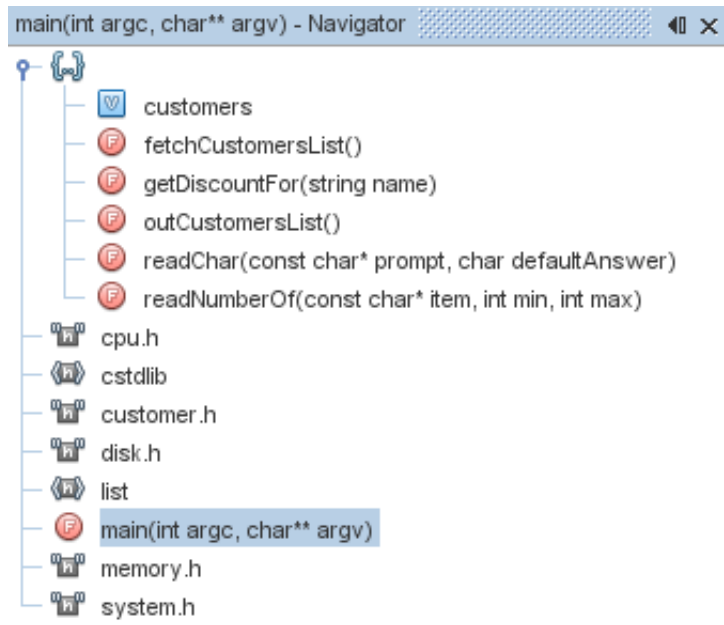


4. 双击 name 变量可打开 customer.h 头文件。

使用 "Navigator" (导航器) 窗口

"Navigator" (导航器) 窗口提供了当前选定文件的紧凑视图，并简化了在文件的不同部分之间的导航。如果 "Navigator" (导航器) 窗口未打开，请选择 "Window" (窗口) > "Navigating" (导航) > "Navigator" (导航器) 打开该窗口。

1. 在 "Editor" (编辑器) 窗口中，单击 quote.cc 文件的任意位置。
2. 在 "Navigator" (导航器) 窗口中会显示该文件的紧凑视图。单击窗口顶部的节点可展开该视图。



3. 要导航到文件的某个元素，请在 "Navigator"（导航器）窗口中双击该元素，"Editor"（编辑器）窗口中的光标将移到该元素。
4. 在 "Navigator"（导航器）窗口中右键单击，可看到用于在窗口中对元素进行排序、对项目进行分组或过滤的选项。
5. 要查看 "Navigator"（导航器）窗口的图标所表示的意义，请通过选择 "Help"（帮助）> "Help Contents"（帮助内容）打开 IDE 联机帮助。在 "Help"（帮助）浏览器中，单击 "Search"（搜索）选项卡，然后在 "Find"（查找）字段中键入导航器图标。

查找类、方法和字段使用情况

您可以使用 "Usages"（使用情况）窗口显示在项目的源代码中使用类（结构）、函数、变量、宏或文件的所有位置。

1. 在 customer.cc 文件中，右键单击第 42 行上的 Customer 类，然后选择 "Find Usages"（查找使用情况）。
2. 在 "Find Usages"（查找使用情况）对话框中，单击 "Find"（查找）。
3. "Usages"（使用情况）窗口将打开，并显示项目的源文件中 Customer 类的所有使用情况。

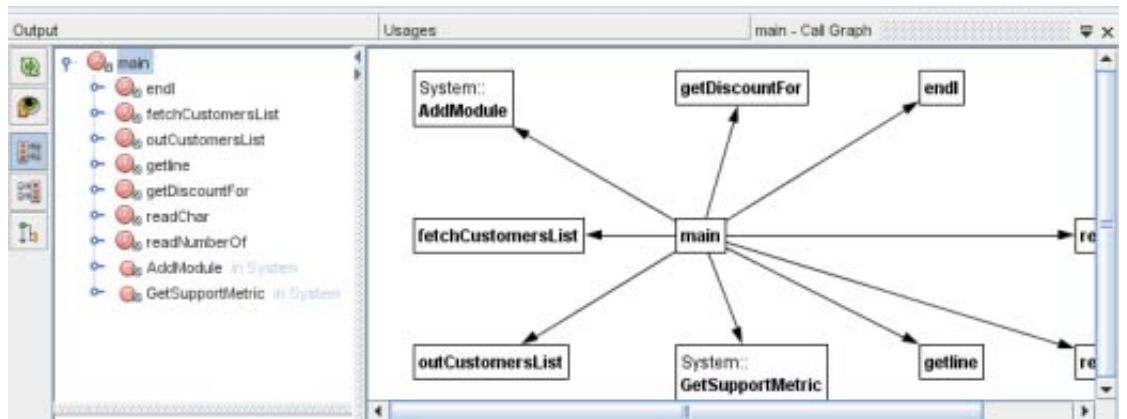


使用调用图

"Call Graph"（调用图）窗口显示类中函数之间的调用关系的两个视图。树视图显示从某个选定函数调用的函数，或者调用该选定函数的函数。图形视图使用箭头来显示被调用函数与调用函数之间的调用关系。

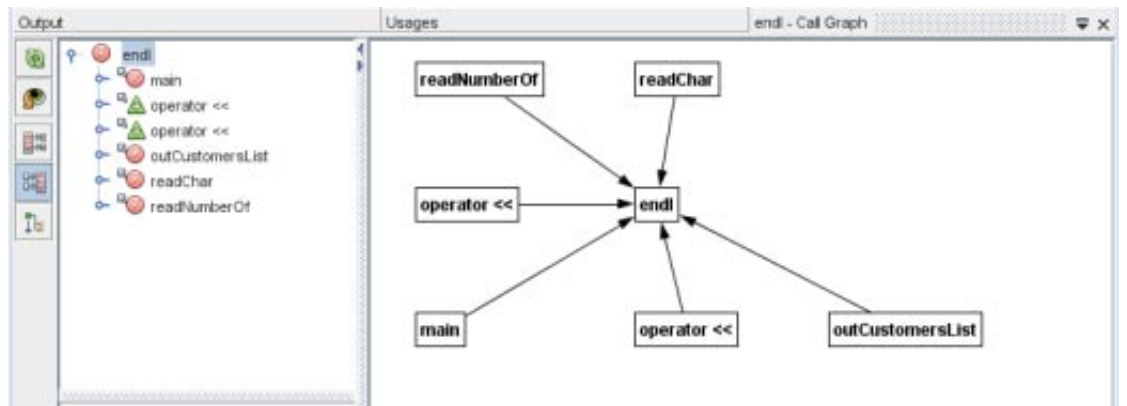
1. 在 quote.cc 文件中，右键单击 main 函数，然后选择 "Show Call Graph"（显示调用图）。

- "Call Graph" (调用图) 窗口将打开，并显示从 main 函数调用的所有函数的树视图和图形视图。

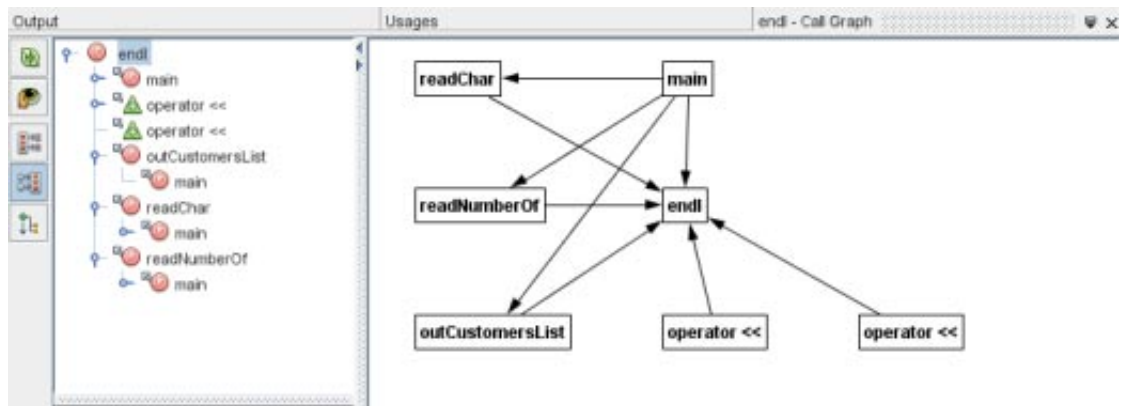


如果屏幕抓图中没有显示所有函数，请单击"Call Graph" (调用图) 窗口左侧的第三个按钮以显示哪个函数是从此函数调用的。

- 展开 endl 节点以显示该函数所调用的函数。请注意图形已经更新，添加了 endl 所调用的函数。
- 选择 endl 节点，单击窗口左侧的第二个按钮以将焦点放到 endl 函数上，然后单击第四个按钮来查看调用 endl 函数的所有函数。



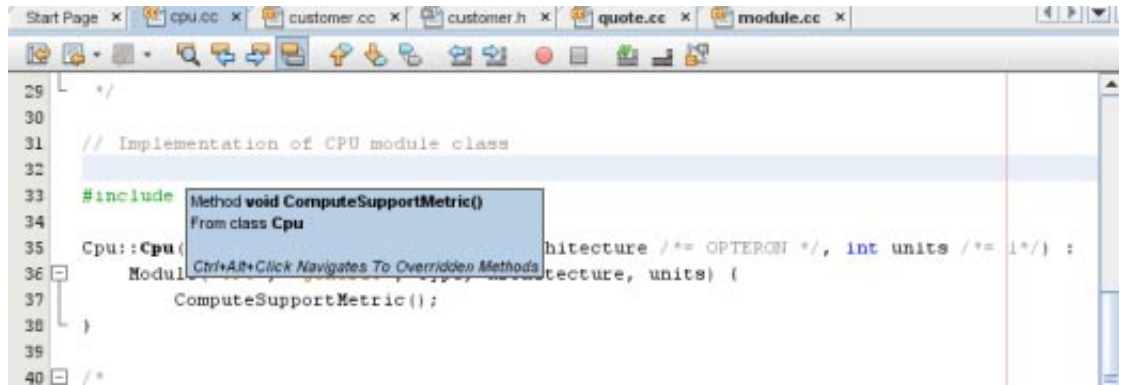
- 展开树中的某些节点可看到更多函数。



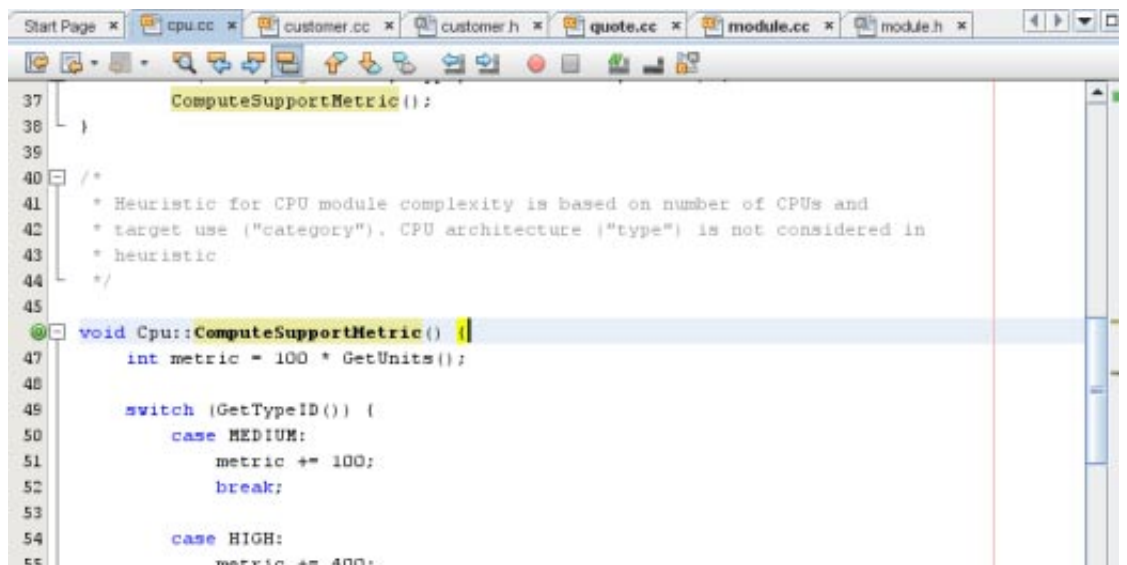
使用超级链接

使用超级链接导航，可以从类、方法、变量或常量的调用跳到声明，也可以从声明跳到定义。使用超级链接，还可以从某个被覆盖的方法跳到覆盖它的方法，反之亦然。

1. 在 Quote_1 项目的 cpu.cc 文件中，在按 Ctrl 键的同时将鼠标移到第 37 行上。将突出显示 ComputeSupportMetric 函数，还会有一条注释显示有关该函数的信息。



2. 单击超级链接，编辑器将跳到函数的定义。



3. 在按住 Ctrl 键的同时将鼠标移到定义上方，然后单击超级链接。编辑器将跳到 cpu.h 头文件中函数的声明。
4. 单击编辑器工具栏中的左箭头，编辑器将跳回到 cpu.cc 中的定义。
5. 将鼠标光标悬停在左边界中的绿色圆圈上方，将看到指示此方法覆盖另一方法的注释。


```

34
35 Cpu::Cpu(int type /*= MEDIUM */, int architecture /*= OPTERON */, int units /*= 1*/) :
36     Module("CPU", "generic", type, architecture, units) {
37         ComputeSupportMetric();
38     }
39
40 /*
41  * Heuristic for CPU module complexity is based on number of CPUs and
42  * target use ("category"). CPU architecture ("type") is not considered in
43  * heuristic
44  */
45 Overrides Module::ComputeSupportMetric
46 void Cpu::ComputeSupportMetric() {
47     int metric = 100 * GetUnits();

```

- 单击绿色圆圈可转到被覆盖的方法，编辑器会跳到 module.h 头文件，该头文件在边界中显示一个灰色圆圈，表示该方法已被覆盖。
- 单击灰色圆圈，编辑器会显示覆盖此方法的方法列表。

```

virtual const char* GetCategory() const = 0;

void SetUnits(int u);
int GetUnits() const;

void SetSupportMetric(int m);
int GetSupportMetric() const;

protected:
virtual void ComputeSupportMetric() = 0; //metric is defined in derived classes
Is Overridden
Cpu::ComputeSupportMetric
Disk::ComputeSupportMetric
Memory::ComputeSupportMetric
int type;

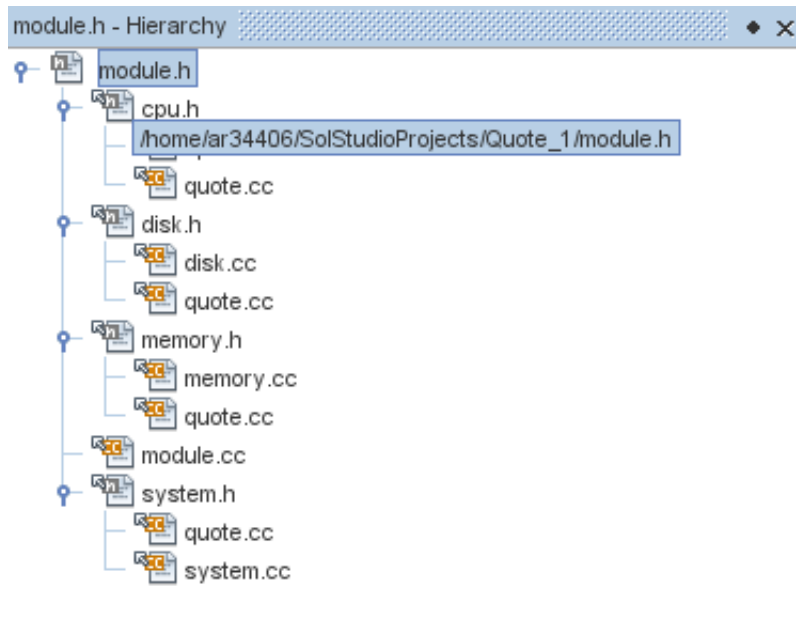
```

- 单击 Cpu::ComputerSupportMetric 项，编辑器会跳回到 cpu.h 头文件中该方法的声明。

使用包含分层结构

使用 "Include Hierarchy" (包含分层结构) 窗口，可以检查直接或间接包含在源文件中的所有头文件和源文件，或者直接或间接包含头文件的所有源文件和头文件。

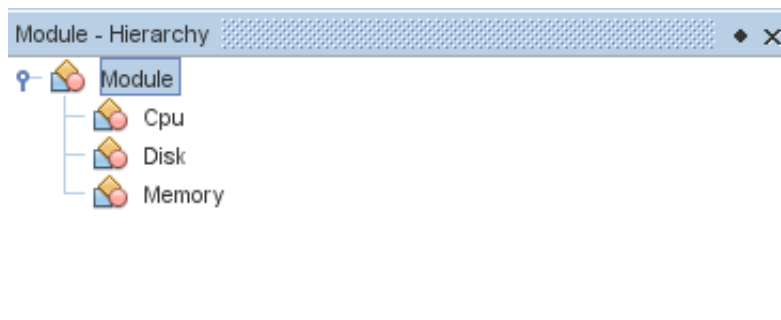
- 在 Quote_1 项目中，在程序编辑器中打开 module.cc 文件。
- 右键单击文件中的 #include "module.h" 行，然后选择 "Navigate" (导航) > "View Includes Hierarchy" (查看包含分层结构)。
- 缺省情况下，"Hierarchy" (分层结构) 窗口显示直接包含头文件的文件的普通列表。单击窗口底部最右侧的按钮可将显示更改为树视图。单击右侧第二个按钮可将显示更改为包含的所有文件或被包含的所有文件。展开树视图中的节点可看到包含头文件的所有源文件。



使用类型分层结构)

使用 "Type Hierarchy" (类型分层结构) 窗口, 可以检查类的所有子类型或父类型。

1. 在 Quote_1 项目中, 打开 module.h 文件。
2. 右键单击 Module 类的声明, 然后选择 "Navigate" (导航) > "View Type Hierarchy" (查看类型分层结构)。
3. "Hierarchy" (分层结构) 窗口将显示 Module 类的所有子类型。

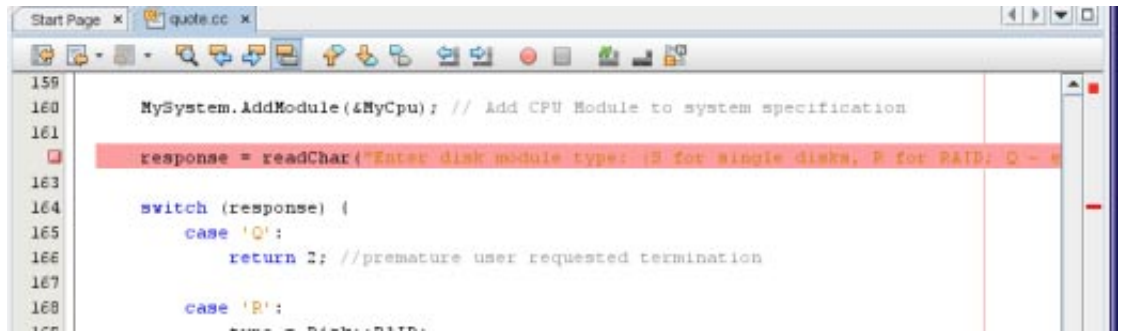


创建断点

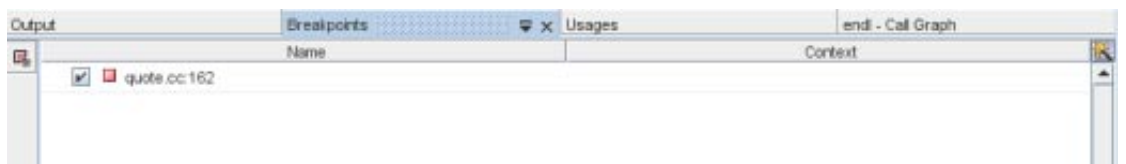
您可以随时创建和处理代码中的断点。

创建和移除行断点

1. 在 Quote_1 项目中, 打开 quote.cc 文件。
2. 通过在 "Editor" (编辑器) 窗口左边界中的第 173 行 (response = readChar("Enter disk module type: (S for single disks, R for RAID; Q - exit)", 'S');) 的旁边单击, 设置一个行断点。该行将以红色突出显示, 表示已设置断点。

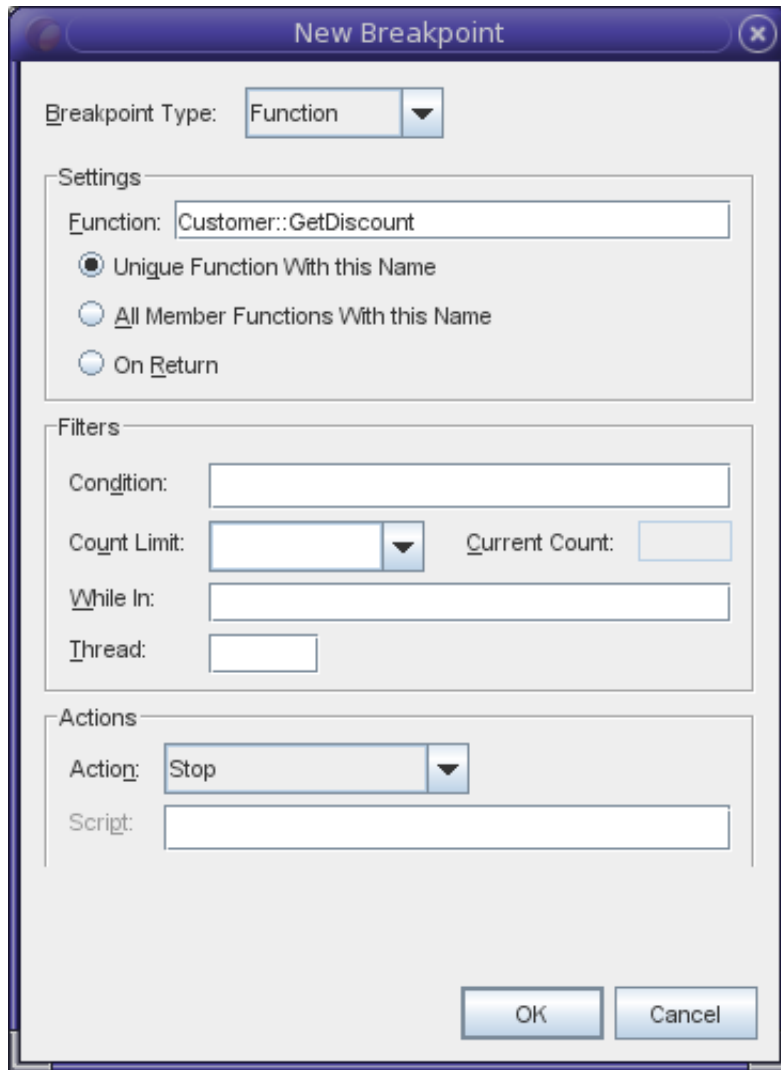


3. 您可以通过单击左边界中的图标移除断点。
4. 选择 "Window" (窗口) > "Debugging" (调试) > "Breakpoints" (断点) 可打开 "Breakpoints" (断点) 窗口。该窗口中会列出您的行断点。

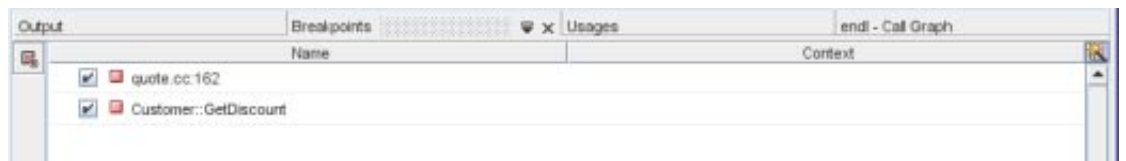


创建函数断点

1. 选择 "Debug" (调试) > "New Breakpoint" (新建断点) (Ctrl+Shift+F8) 可打开 "New Breakpoint" (新建断点) 对话框。
2. 在 "Breakpoint Type" (断点类型) 下拉式列表中, 将类型设置为 "Function" (函数)。
3. 在 "Function" (函数) 文本字段中, 键入函数名称 Customer::GetDiscount。单击 "OK" (确定)。



4. 您的函数断点现已设置完毕，并添加到了"Breakpoints"（断点）窗口中的列表中。



调试项目

启动调试会话时，IDE 会启动 dbx 调试器，然后在调试器内部运行应用程序。IDE 会自动打开调试器窗口，并将调试器输出内容输出到"Output"（输出）窗口。

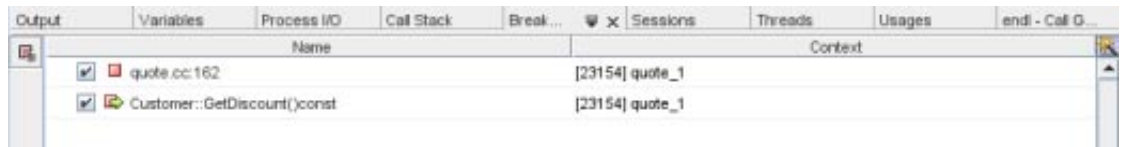
启动调试会话

1. 通过右键单击项目节点然后选择"Debug"（调试），可为 Quote_1 项目启动一个调试会话。调试器将启动，应用程序开始运行，并在"Debugging"（调试）窗口中打开"Variables"（变量）、"Watches"（监视）、"Call Stack"（调用栈）、"Threads"（线程）、"Process I/O"（进程 I/O）和"Dbx Console"（Dbx 控制台）窗口。
2. 通过选择"Window"（窗口）>"Debugging"（调试）>"Sessions"（会话）可打开"Sessions"（会话）窗口。调试会话将显示在此窗口中。

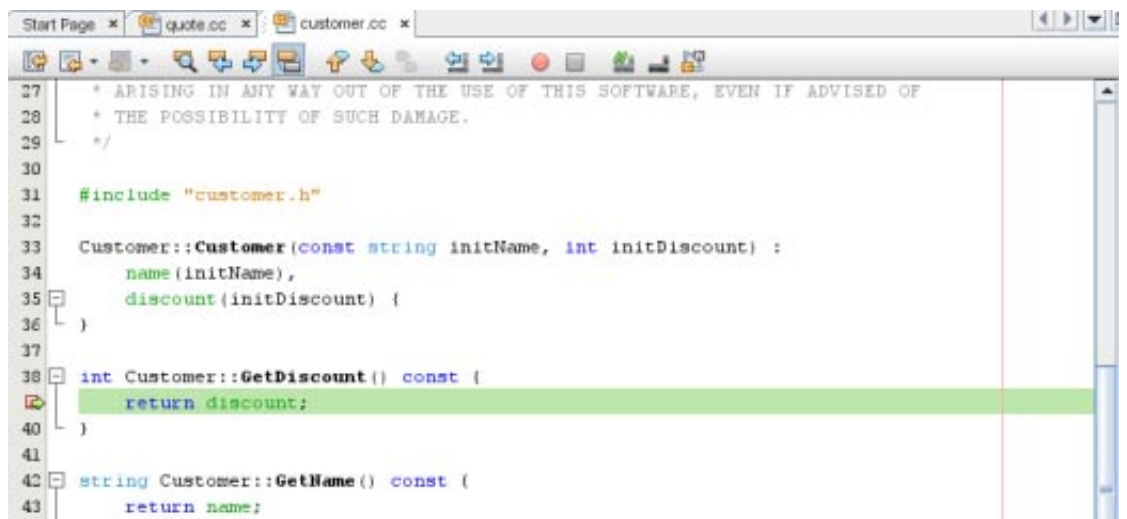


检查应用程序的状态

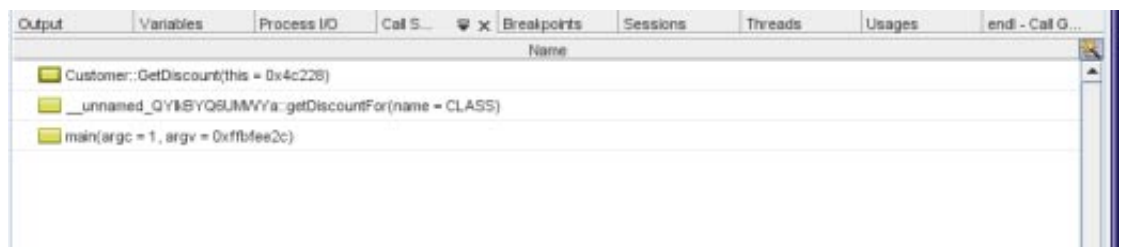
1. Quote_1 应用程序提示您在 "Process I/O" (进程 I/O) 窗口中进行输入。
2. 在 Enter customer name: 提示符后输入客户名称。
3. 应用程序将在先前设置的函数断点处停止。"Breakpoints" (断点) 窗口列出了先前设置的两个断点。绿色程序计数器箭头显示在函数断点的断点图标顶部。



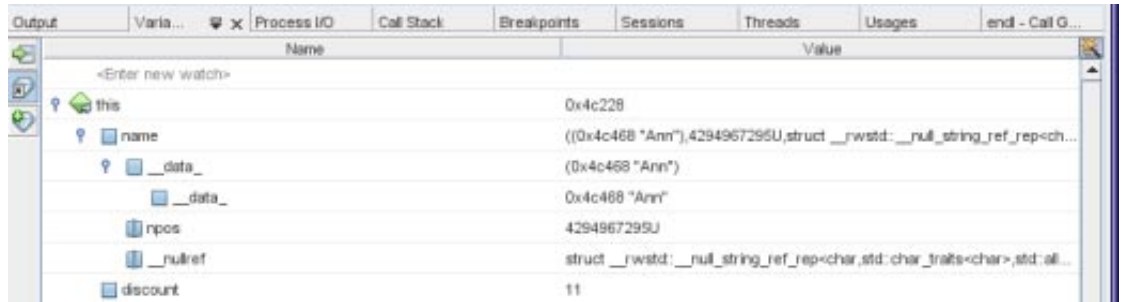
4. 在 customer.cc 文件中，绿色程序计数器箭头显示在 GetDiscount 函数第一行上的断点图标顶部。



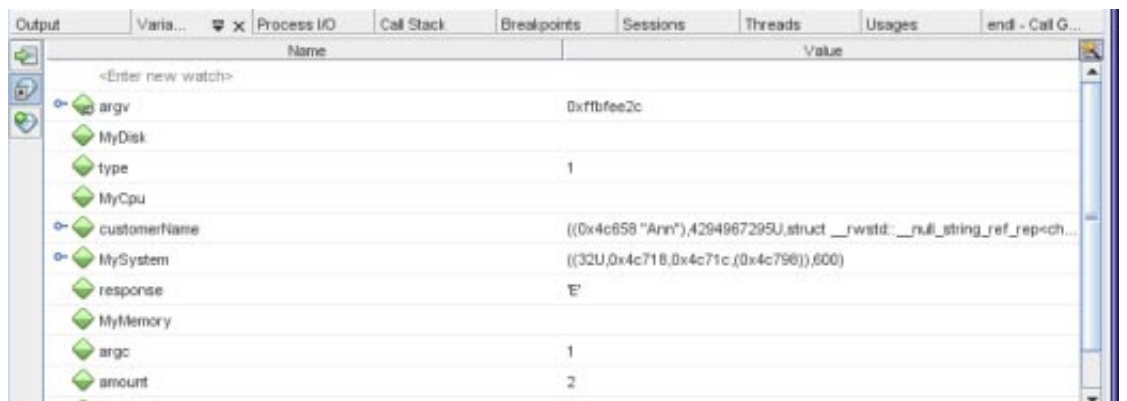
5. 单击 "Call Stack" (调用栈) 窗口。调用栈将显示三个帧。



6. 单击 "Variables" (变量) 窗口，请注意会显示一个变量。单击节点可展开结构。



- 单击 "Continue" (继续) 按钮。执行 GetDiscount 函数，将客户折扣输出到 "Process I/O" (进程 I/O) 窗口，然后系统会提示您进行输入。
- 请根据提示输入内容。程序会在下一个断点 (先前设置的行断点) 处停止。单击 "Variables" (变量) 窗口，注意长长的局部变量列表。



- 单击 "Call Stack" (调用栈) 窗口，注意栈中只有一个帧。
- 单击 "Continue" (继续) 按钮，根据 "Process I/O" (进程 I/O) 窗口中的提示继续输入内容，直到程序完成。将最后的输入内容输入到程序中后，调试会话将结束。要在程序执行完成之前结束调试会话，可以在 "Sessions" (会话) 窗口中右键单击该会话，然后选择 "Finish" (完成)。

在机器指令级调试

调试器提供了用于在机器指令级调试项目的窗口。

- 右键单击 Quote_1 项目，然后选择 "Debug" (调试)。
- 在 "Process I/O" (进程 I/O) 窗口中，根据提示输入客户名称。
- 当程序在 GetDiscount 函数上的断点处暂停时，选择 "Window" (窗口) > "Debugging" (调试) > "Disassembly" (反汇编) 可打开 "Disassembly" (反汇编) 窗口，如同在 "Editor" (编辑器) 窗口中一样。绿色程序计数器箭头将显示在暂停程序的指令处的断点图标顶部。

```

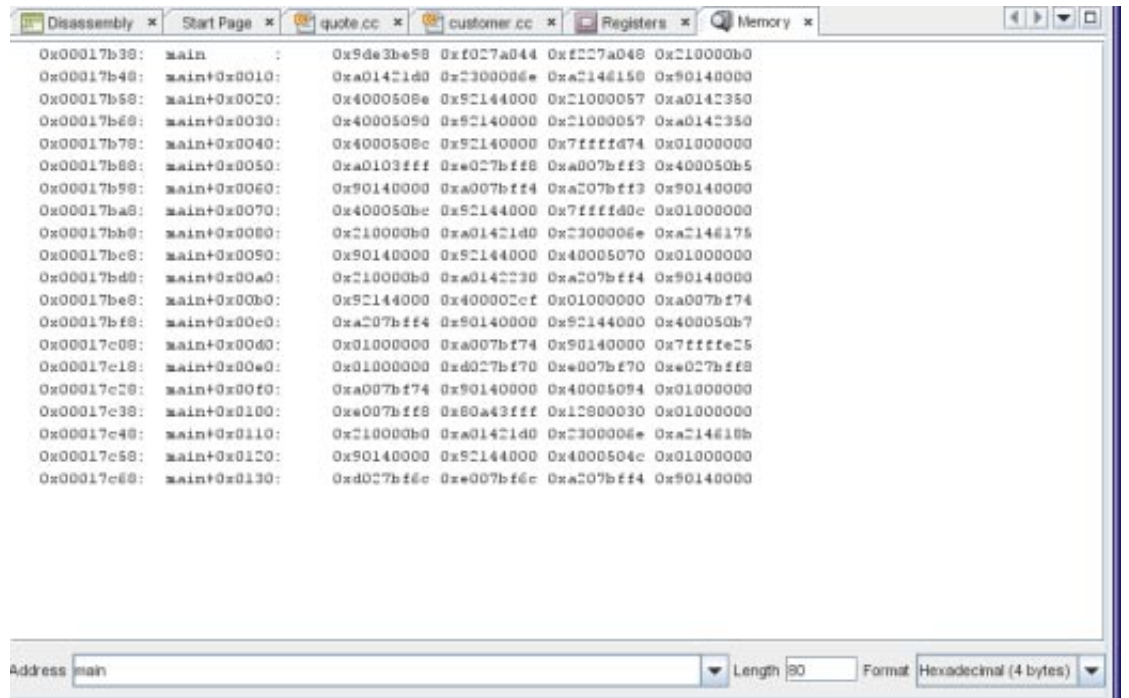
0x0001a60c: Customer+0x000c: 02      %10, %g0, %g0
0x0001a610: Customer+0x0040: call   ex_rethrow_q [PLT]    ! 0x2bd60
0x0001a614: Customer+0x0044: nop
0x0001a618: Customer+0x0048: ret
0x0001a61c: Customer+0x004c: restore
36   }
37
38   int Customer::GetDiscount() const {
39       return discount;
0x0001a630: GetDiscount    :      save   %sp, -104, %sp
0x0001a634: GetDiscount+0x0004:      st    %10, [%fp + 68]
0x0001a638: GetDiscount+0x0008:      ld    [%fp + 68], %10
0x0001a63c: GetDiscount+0x000c:      ld    [%10 + 4], %10
0x0001a640: GetDiscount+0x0010:      st    %10, [%fp - 4]
0x0001a644: GetDiscount+0x0014:      ld    [%fp - 4], %10
0x0001a648: GetDiscount+0x0018:      or    %10, %g0, %10
0x0001a64c: GetDiscount+0x001c:      ret
0x0001a650: GetDiscount+0x0020:      restore
40   }
41
42   string Customer::GetName() const {
43       return name;

```

4. 选择 "Window" (窗口) > "Debugging" (调试) > "Registers" (寄存器) 可打开 "Registers" (寄存器) 窗口，该窗口显示寄存器的内容。

Name	Value
g0-g1	0x00000000 0x00000000 0x00000000 0x00000000
g2-g3	0x00000000 0x00000010 0x00000000 0xff366798
g4-g5	0x00000000 0x00000000 0x00000000 0xfefeeaa0
g6-g7	0x00000000 0x00000000 0x00000000 0xff1f2a80
o0-o1	0x00000000 0x00000000 0x00000000 0x00000001
o2-o3	0x00000000 0x00000001 0x00000000 0x00000000
o4-o5	0x00000000 0x0000000e 0x00000000 0x0000000e
o6-o7	0x00000000 0xffbfab90 0x00000000 0xfefcc7a8
i0-i1	0x00000000 0x0004c228 0x00000000 0x0004c228
i2-i3	0x00000000 0x0004c468 0x00000000 0x00000001
i4-i5	0x00000000 0xfefeea78 0x00000000 0x0004c440
i6-i7	0x00000000 0xfefee4b4 0x00000000 0x00000040
i0-i1	0x00000000 0x0004c228 0x00000000 0x00000001
i2-i3	0x00000000 0x00021d36 0x00000000 0x00000000
i4-i5	0x00000000 0x00000001 0x00000000 0xff36452c
i6-i7	0x00000000 0xffbfab90 0x00000000 0x00017584
y	0x00000000 0x00000000
ect	0x00000000 0x00000000
pc	0x00000000 0x0001a638: GetDiscount+0x8 ld [%fp + 68], %10
npc	0x00000000 0x0001a63c: GetDiscount+0xc ld [%10 + 4], %10
f0f1	+0.00000000000000e+00
f2f3	+0.00000000000000e+00
f4f5	+0.00000000000000e+00
f6f7	-NaN
f8f9	-NaN
f10f11	-NaN
f12f13	-NaN
f14f15	-NaN

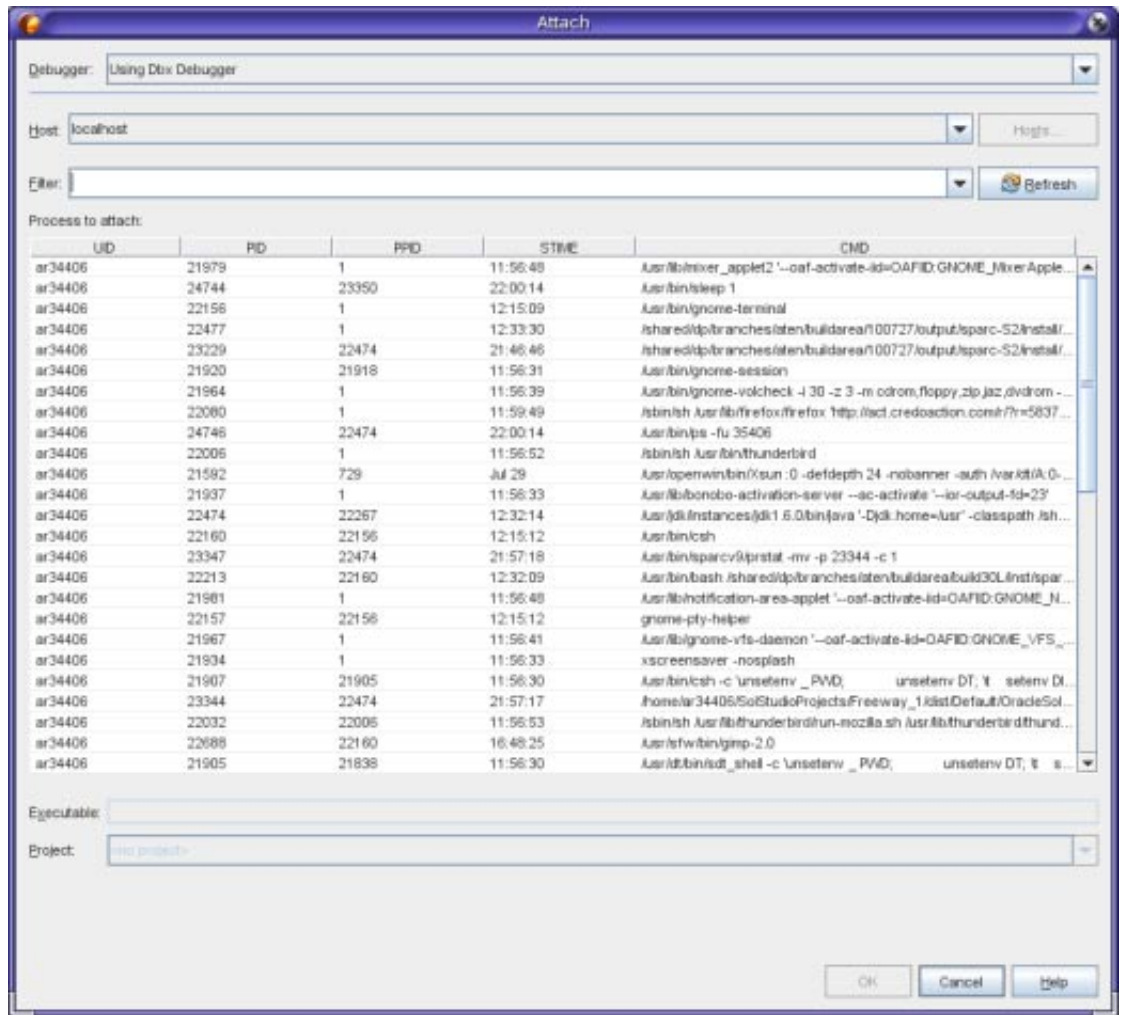
5. 选择 "Window" (窗口) > "Debugging" (调试) > "Memory" (内存) 可打开 "Memory" (内存) 窗口，该窗口显示项目当前使用内存的内容。在窗口底部，可以指定要浏览的内存地址、更改内存浏览的长度，或者更改内存信息的格式。



通过附加到某个正在运行的程序对其进行调试

如果要调试某个已在运行的程序，可以将调试器附加到相应的进程。

1. 选择 "File" (文件) > "New Project" (新建项目)。
2. 在 "New Project" (新建项目) 向导中，展开 "Samples" (样例) 节点，然后选择 "C/C++" 类别。
3. 选择 "Freeway Simulator" (Freeway 仿真器) 项目。单击 "Next" (下一步)，然后单击 "Finish" (完成)。
4. 右键单击所创建的 Freeway_1 项目，然后选择 "Run" (运行)。项目现已生成，"Freeway" 应用程序将启动。
5. 选择 "Debug" (调试) > "Attach Debugger" (附加调试器)。

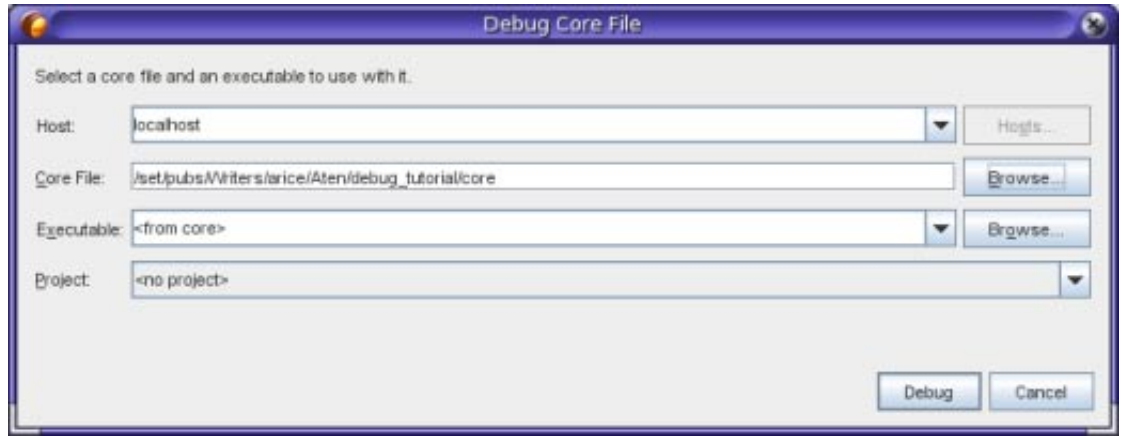


6. 在 "Attach" (附加) 对话框中, 在 "Filter" (过滤器) 字段中键入 Freeway 以过滤处理器列表。
7. 从过滤后的列表中选择 Freeway 进程。
8. 单击 "OK" (确定)。
9. 将启动一个调试会话, Freeway 进程的执行会在调试器附加到该进程的位置暂停。

调试核心转储文件

如果程序即将崩溃, 您可能需要调试核心转储文件 (程序崩溃时的内存映像)。将核心转储文件装入调试器:

1. 选择 "Debug" (调试) > "Debug core file" (调试核心转储文件)。
2. 在 "Debug core file" (调试核心转储文件) 字段中键入核心转储文件的完整路径, 或者在 "Select Core File" (选择核心转储文件) 对话框中单击 "Browse" (浏览) 并导航到您的核心转储文件。



3. 如果调试器无法将您指定的核心转储文件与某个可执行文件相关联，"Select Executable"（选择可执行文件）对话框将打开，以便您指定可执行文件。如果出现这种情况，请在"Executable"（可执行文件）文本框中键入可执行文件的路径名，或者单击"Browse"（浏览）按钮，然后使用"Executable"（可执行文件）对话框选择可执行文件。
4. 缺省情况下，"Project"（项目）文本字段会显示 <no project> 或者与可执行文件名称完全匹配的某个现有项目的名称。如果需要为可执行文件创建一个新项目，请选择 <create new project>。
5. 单击"Debug"（调试）。

要获取有关调试的更加深入的教程，请参见《[Oracle Solaris Studio 12.2 dbxtool 教程](#)》。

版权所有 ©2010 本软件和相关文档是根据许可证协议提供的，该许可证协议中规定了关于使用和公开本软件和相关文档的各种限制，并受知识产权法的保护。除非在许可证协议中明确许可或适用法律明确授权，否则不得以任何形式、任何方式使用、拷贝、复制、翻译、广播、修改、授权、传播、分发、展示、执行、发布或显示本软件和相关文档的任何部分。除非法律要求实现互操作，否则严禁对本软件进行逆向工程设计、反汇编或反编译。

此文档所含信息可能随时被修改，恕不另行通知，我们不保证该信息没有错误。如果贵方发现任何问题，请书面通知我们。

如果将本软件或相关文档交付给美国政府，或者交付给以美国政府名义获得许可证的任何机构，必须符合以下规定：

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

本软件或硬件是为了在各种信息管理应用领域内的一般使用而开发的。它不应被应用于任何存在危险或潜在危险的应用领域，也不是为此而开发的，其中包括可能会产生人身伤害的应用领域。如果在危险应用领域内使用本软件或硬件，贵方应负责采取所有适当的防范措施，包括备份、冗余和其它确保安全使用本软件或硬件的措施。对于因在危险应用领域内使用本软件或硬件所造成的一切损失或损害，Oracle Corporation 及其附属公司概不负责。

Oracle 和 Java 是 Oracle 和/或其附属公司的注册商标。其他名称可能是各自所有者的商标。

AMD、Opteron、AMD 徽标以及 AMD Opteron 徽标是 Advanced Micro Devices 的商标或注册商标。Intel 和 Intel Xeon 是 Intel Corporation 的商标或注册商标。所有 SPARC 商标均是 SPARC International, Inc 的商标或注册商标，并应按照许可证的规定使用。UNIX 是通过 X/Open Company, Ltd 授权的注册商标。

本软件或硬件以及文档可能提供了访问第三方内容、产品和服务的方式或有关这些内容、产品和服务的信息。对于第三方内容、产品和服务，Oracle Corporation 及其附属公司明确表示不承担任何种类的担保，亦不对其承担任何责任。对于因访问或使用第三方内容、产品或服务所造成的任何损失、成本或损害，Oracle Corporation 及其附属公司概不负责。

821-2515

Oracle Corporation 500 Oracle Parkway, Redwood City, CA 94065 U.S.A.

