Oracle® Solaris Studio 12.4：IDE 快速入门教程

文件号码 E57223
2014 年 10 月

ORACLE

目录

♦ ♦ ♦ 第 1 章

Oracle Solaris Studio IDE 概述

本章介绍了使用 Oracle Solaris Studio IDE 的优点。

为什么要使用 Oracle Solaris Studio IDE？

Oracle Solaris Studio 提供了一个基于 NetBeans 平台构建并使用 Oracle Solaris Studio 工具套件直接在环境中配置的图形化集成开发环境 (Integrated Development Environment, IDE)。在 IDE 中，可以执行以下操作：

- 使用 **Oracle Solaris Studio C、C++ 和 Fortran 编译器**以及 `dmake` **分布式 make 命令**编译和生成代码。
- 使用 `dbx` **调试器**调试代码。有关更多信息，请参见第 7 章 调试应用程序。
- 使用集成了 Oracle Solaris Studio 分析工具的分析工具分析代码：
  - **“运行监视器”工具**使用性能分析器的数据收集功能。
  - **“内存分析”工具**使用 `discover's` 动态内存检查功能。有关更多信息，请参见“对项目运行内存访问检查” [81]。
  - **“静态代码检查”**使用代码分析器的静态错误数据收集功能。有关更多信息，请参见“使用静态代码错误检查” [34]。
  - **“数据争用和死锁”工具**使用线程分析器的数据收集和分析功能。
- 当使用**远程开发**在远程服务器上安装 Oracle Solaris Studio 编译器和工具时，在本地运行 IDE。有关更多信息，请参见“执行远程开发” [85]。
- 当 IDE 使用**远程桌面分发**在远程生成服务器上运行时，创建 IDE 的特殊分发。

本指南其余部分介绍了如何使用 IDE 中的所有这些功能。

♦ ♦ ♦ 第 2 章

2 创建新应用程序

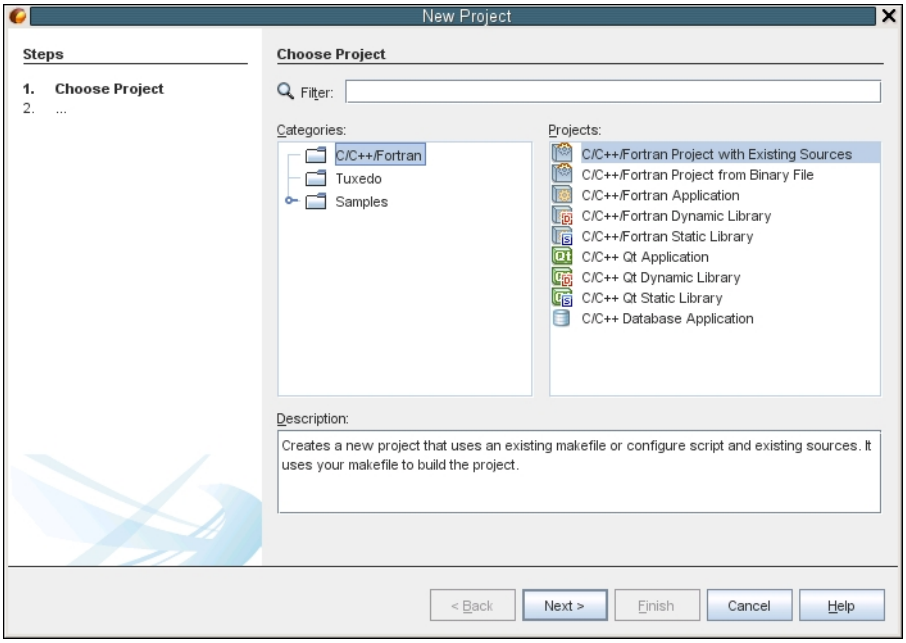
本章介绍了在 IDE 中创建新应用程序项目（有时称为受管项目）的步骤。有关更多信息，请参见“创建应用程序项目” [9]。

要了解如何基于现有源代码创建项目，请参见第 4 章 基于现有源代码创建项目。

本章还介绍了可以对项目执行的其他任务，如下所示：

创建应用程序项目

1. 通过选择 "File"（文件）> "New Project"（新建项目）(Ctrl+Shift+N)，打开 "New Project"（新建项目）向导。
2. 在向导中选择 "C/C++/Fortran" 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。

将创建一个有多个逻辑文件夹的项目，这些文件夹显示在 "Projects"（项目）窗口中：

- Source Files（源文件）
- Header Files（头文件）
- Resource Files（资源文件）
- Test Files（测试文件）
- Important Files（重要文件）

基于现有源代码创建 C、C++ 或 Fortran 项目时会同时创建一个逻辑文件夹 "Important Files"（重要文件）。

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 "Source Files"（源文件）文件夹、"Header Files"（头文件）文件夹和其他逻辑文件夹的文件将成为项目的一部分，并且在生成项目时进行编译。

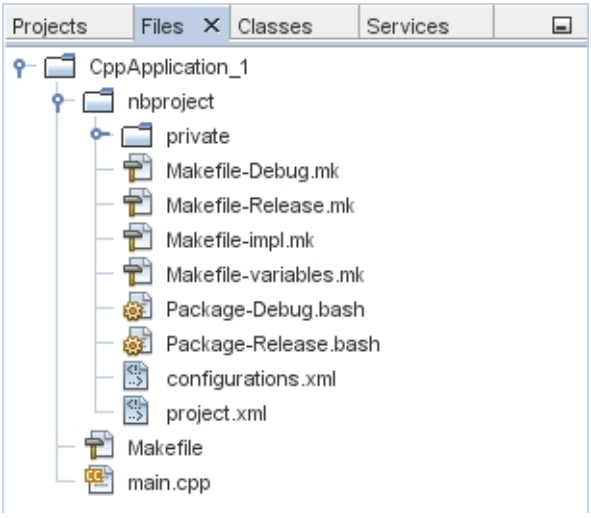
注 - 如果要将添加到 "Header Files"（头文件）文件夹的头文件编译到程序中，则必须采用正常方式通过 `#include` 语句在源文件中引用这些头文件。

添加到 "Important Files"（重要文件）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。

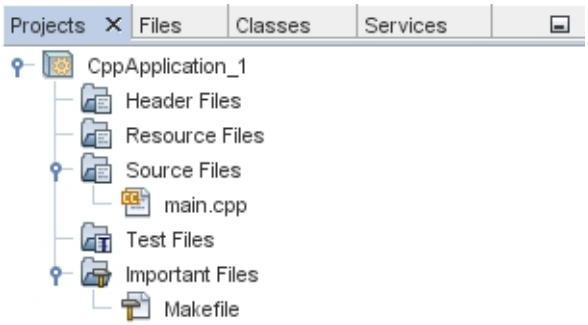
在项目的逻辑视图和物理视图之间切换

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

以下各节介绍了如何通过创建或者引入现有源代码，将不同类型的文件和文件夹添加到当前项目中。

为项目创建逻辑文件和文件夹

您可以创建逻辑文件夹并将其添加到项目中。

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。

为项目创建新的源文件

您可以创建新的源文件并将其添加到项目中。

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

注 - 还可以从此文件夹中删除文件。在本例中，您不需要在创建项目时缺省添加的 `main.cpp` 文件。要从项目中删除此文件，请右键单击此文件名称，然后选择 "Remove From Project"（从项目中删除）。

为项目创建头文件

您可以创建新的头文件并将其添加到项目中。

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"（源文件）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"（源文件）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。

设置属性和配置

以下各节介绍了如何设置和管理当前项目及各个文件的属性。

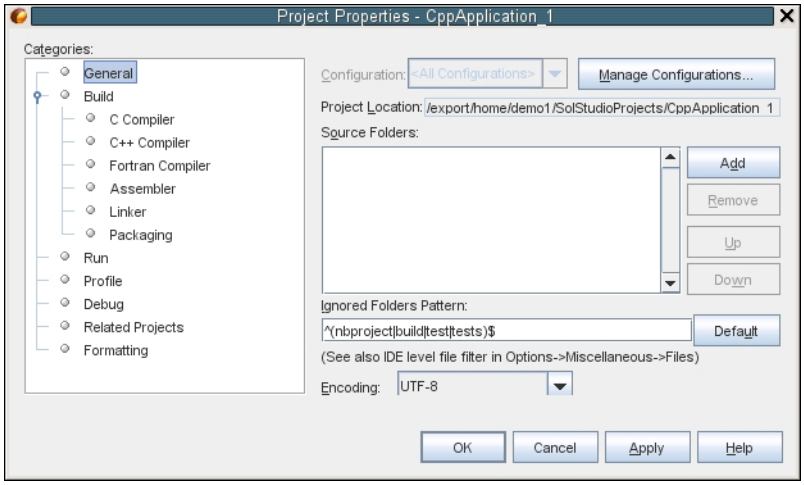
- “设置项目属性” [13]
- “管理配置” [14]
- “设置源文件属性” [15]

设置项目属性

在创建项目时，有两个配置："Debug"（调试）和 "Release"（发行）。配置是指项目使用的一组设置，可用于轻松地一次性切换许多属性设置。"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

创建项目后，它将具有 "Debug"（调试）配置和 "Release"（发行）配置。配置是用于项目的设置集合。当您选择配置时，可以轻松地一次切换多项设置。"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。

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。

提示 - 您也可以在 IDE 的工具栏中设置活动配置，或者右键单击项目节点，然后选择 "Set Configuration"（设置配置）。

设置源文件属性

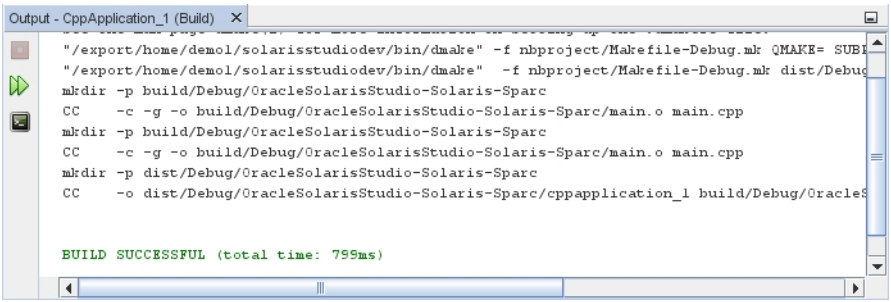
设置项目的项目属性时，相关属性将应用到项目中的所有文件。可以为某个特定文件设置一些属性。

1. 右键单击 `newmain.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。
3. 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。
4. 取消 "Project Properties"（项目属性）对话框。

生成项目

生成项目：

1. 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。



2. 在主工具栏的配置下拉式列表中，将配置由 "Debug" 切换为 "Performance Release"。现在，将使用 "Performance Release" 配置来生成项目。
3. 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目标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 `newmain.c` 文件，然后选择 "Compile File"（编译文件）。将仅编译此文件。

♦ ♦ ♦ 第 3 章

3 运行项目

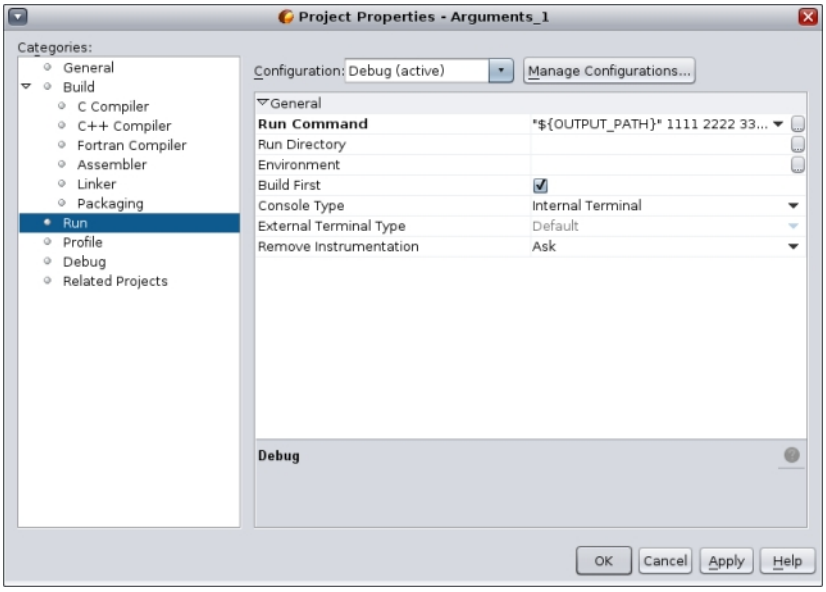
本章介绍了如何在 IDE 中运行项目，包含以下各节：

运行项目

`Arguments` 样例程序会输出命令行参数。在运行该程序之前，您将在当前配置中设置一些参数。然后，您将运行该程序。

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 "Run Command"（运行命令）文本字段中，在输出路径后面键入 `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。

启动器

您可以创建“启动器”，以便使用不同的参数从项目上下文菜单轻松地运行项目，或者从脚本启动项目。有关启动器的更多信息，请参见《Oracle Solaris Studio 12.4 新增功能》中的“IDE 中的新启动器功能”。

♦ ♦ ♦ 第 4 章

基于现有源代码创建项目

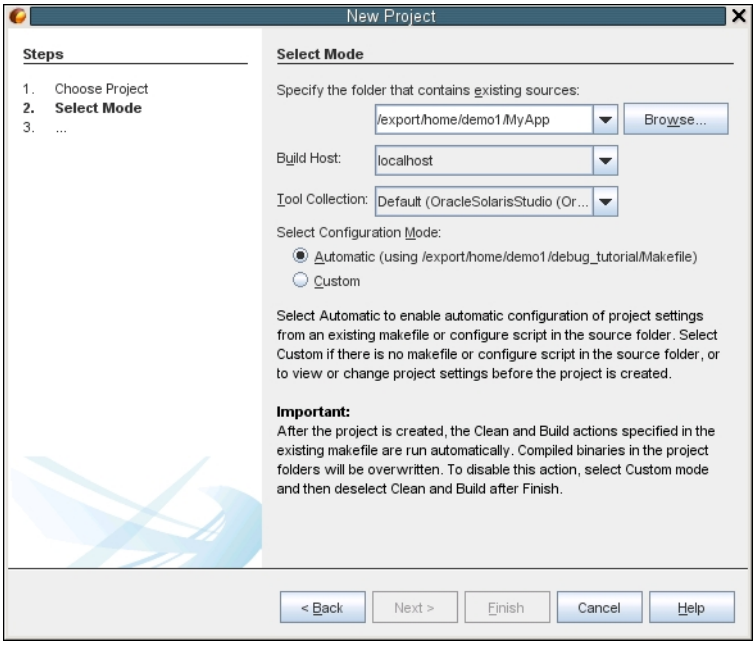
您也可以从已经包含 makefile 的现有源文件，或者在您运行 `configure` 脚本时构建 makefile 的现有源文件创建项目。这种类型的项目称为非受管项目。有关项目和 makefile 的更多信息，请参见 IDE 帮助中的 "C/C++/Fortran Projects and Makefiles"（C/C++/Fortran 项目和 Makefile）页。

本章包含以下各节：

基于现有源代码创建项目

对于 "C/C++/Fortran Project From Existing Sources"（基于现有源代码的 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++/Fortran" 类别。
3. 选择 "C/C++/Fortran Project From Existing Sources"（基于现有源代码的 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）按钮。



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。

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）。

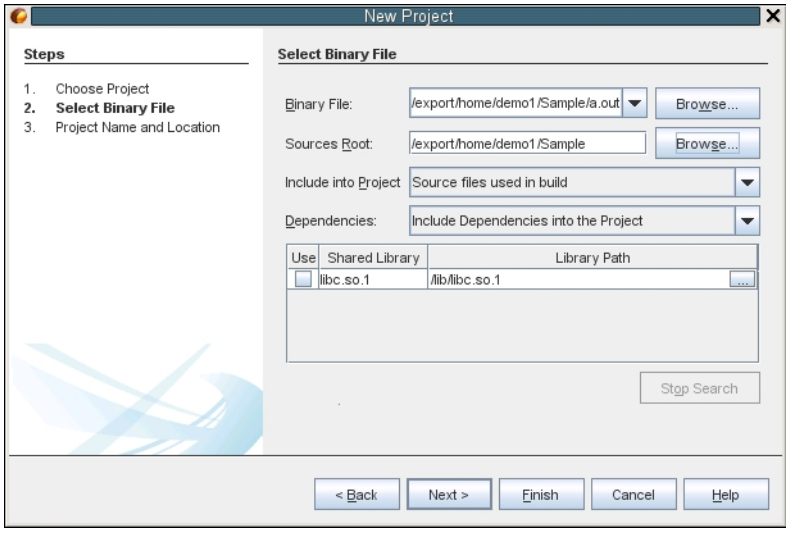
基于二进制文件创建项目

对于 "C/C++/Fortran project from a binary file"（基于二进制文件的 C/C++/Fortran 项目），您可以通过多种方式基于现有的二进制文件创建项目。

通过 "New Project"（新建项目）向导基于二进制文件创建项目

1. 选择 "File"（文件）> "New Project"（新建项目）。
2. 选择 "C/C++/Fortran" 类别。
3. 选择 "C/C++/Fortran Project from Binary File"（基于二进制文件的 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 Binary File"（选择二进制文件）页面上，单击 "Browse"（浏览）按钮。在 "Select Binary File"（选择二进制文件）对话框中，导航到要基于其创建项目的二进制文件。

   生成二进制文件时所基于的源文件的根目录是自动填充的。缺省情况下，项目中只包括生成二进制文件时所基于的源文件。缺省情况下，项目中包括相关项。项目所需的共享库会自动列出。

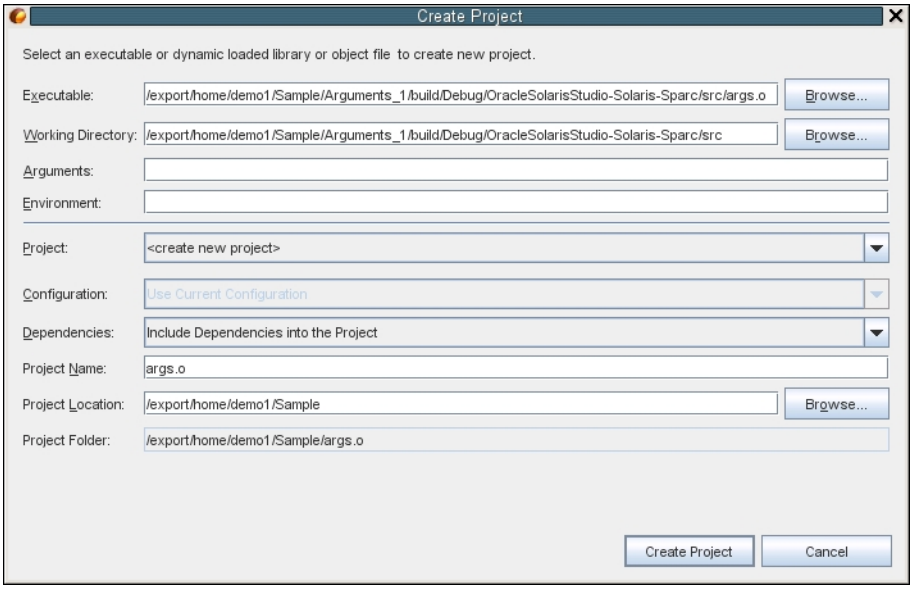


5. 单击 "Next"（下一步）。
6. 在 "Project Name and Location"（项目名称和位置）页面上，可以选择项目的名称和位置。单击 "Finish"（完成）。

在 "Files"（文件）窗口中基于二进制文件创建项目

1. 在 "Files"（文件）窗口中，转至 Arguments_1 项目并展开生成节点，一直往下到 `args.o`。
2. 执行下列操作之一以显示 "Create Project"（创建项目）向导。

- 右键单击 `args.o` 文件并单击 "Create Project..." （创建项目...）
- 将 `args.o` 文件拖动到源代码编辑器中。



将弹出 "Create Project"（创建项目）向导，其中已预填充 "Executable"（可执行文件）和 "Working Directory"（工作目录）。

3. 输入您运行项目时要使用的任何参数和环境变量。在 "Project"（项目）下拉菜单中，选择 "<create new project>"（<创建新项目>）。

   如果单击右下角的 "Create Project"（创建项目）按钮，将会在项目选项卡中创建一个名为 Args.o 的新项目。

4. 单击 "Cancel"（取消）。

代码帮助

代码帮助是一组 IDE 功能，可帮助您导航和编辑源代码，尤其是对于非受管项目。有关编辑和导航的更多信息，请参见第 6 章 编辑和导航源文件。

对于非受管项目，您可以指定如何解析代码以启用 IDE 的代码帮助功能。本节将讨论如何配置代码帮助和代码帮助高速缓存共享。

配置代码帮助

创建项目时，您可以指定可供内置解析器用来创建配置的包含文件和宏定义。或者，如果构建的项目具有调试信息，您可以让内置解析器自动搜索编译的每个源文件的包含文件和宏定义。

要指定可提高项目的 IDE 代码帮助功能准确性的其他代码帮助配置信息，请使用 "Configure Code Assistance"（配置代码帮助）向导。要启动该向导，请右键单击项目，然后选择 "Configure Code Assistance"（配置代码帮助）。要了解有关配置代码帮助和 "Configure Code Assistance"（配置代码帮助）向导的更多信息，请参见 IDE 中的相关帮助部分。

共享代码帮助高速缓存

解析 C/C++ 源代码时，IDE 会把解析结果存储到磁盘上的代码帮助高速缓存中。打开项目时，IDE 将检查高速缓存是否为最新的。如果高速缓存是最新的，则 IDE 不会解析项目，而仅会加载从代码帮助高速缓存导航代码所需的数据。

缺省情况下，IDE 将在 `$`*userdir*`/var/cahche` 文件夹下您的用户目录中为所有项目创建一个代码帮助高速缓存。用户目录中的高速缓存不能复制或共享到其他位置。

但是，如果代码帮助高速缓存位于项目内，当某计算机的操作系统与解析代码所在的操作系统相同，且项目所用的工具集合在该计算机的同一位置可用时，可以将代码帮助高速缓存复制到该计算机。

有关代码帮助高速缓存共享的更多信息以及要了解如何指示 IDE 将代码帮助高速缓存置于项目元数据内，请参见《Oracle Solaris Studio 12.4 新增功能 》中的“共享代码帮助高速缓存”以及 IDE 帮助中的 "Relocating the Code Assistance Cache for Version Controlled Projects"（为版本控制的项目重定位代码帮助高速缓存）。

代码帮助的项目属性选项

IDE 目前提供了以下项目属性，使您可以更方便地在版本控制系统中使用非受管项目。

瞬态宏	您可以提供可变的宏（`-D` 选项）列表。它们取决于时间、日期或特定环境。这些宏环境变量值不会随项目的公共元数据一起存储。
用户环境变量	您可以提供环境变量列表，项目使用这些环境变量来传递特定于系统的路径。这些宏环境变量值不会随项目的公共元数据一起存储。对于非受管项目，您可以指定存储项目元数据时要使用的环境变量列表。当 IDE 存储编译器选项并且选项值与变量值一致时，将改为编写宏。

搜索文件系统以查找 C/C++ 头文件

如果创建其源代码尚未生成并且不包含任何调试信息的非受管项目，则在配置代码帮助时 IDE 可能会遇到问题。在这种情况下，您可以指定在 "Configure Code Assistance"（配置代码帮助）向导中使用一种特殊的模式，即在文件系统中搜索 C/C++ 头文件。在这种模式下，IDE 会尝试在文件系统中搜索头文件，以解析执行失败的 include 指令。该向导会让您输入头文件的搜索路径。缺省情况下，该路径是项目的源根目录。

♦ ♦ ♦ 第 5 章

创建 Oracle 数据库项目

本章介绍了如何在 IDE 中创建 Oracle 数据库项目，并指向多个参考资料以了解有关 Oracle 数据库项目以及如何在 IDE 中使用这些项目的更多信息。有关 Oracle 数据库项目和 IDE 的更多资源，请参见“Oracle 数据库项目” [93]。

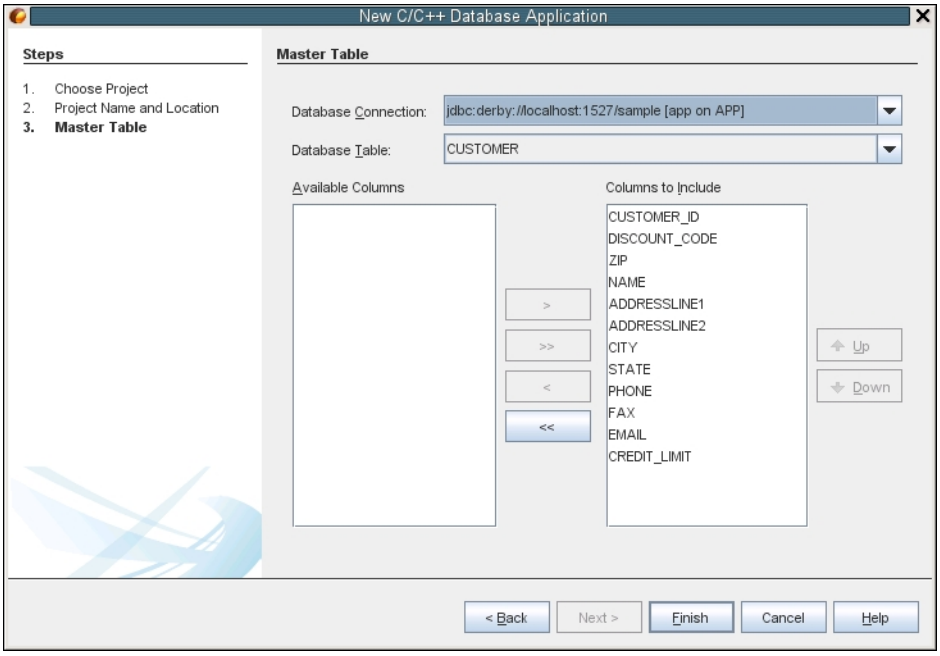
创建 Oracle 数据库项目

您可以为 Oracle 数据库应用程序创建项目。要执行此操作，您使用的 Oracle Solaris Studio 安装必须包括可选的 Oracle Instant Client 组件。

1. 选择 "File"（文件）> "New Project"（新建项目）。
2. 在 "New Project"（新建项目）对话框中，选择 "C/C++/Fortran" 类别和 "C/C++ Database Application"（C/C++ 数据库应用程序）项目。单击 "Next"（下一步）。
3. 在 "Project Name and Location"（项目名称和位置）页面上，可以选择项目的名称和位置。单击 "Next"（下一步）。
4. 在 "Master table"（主表）页面上，从 "Database Connection"（数据库连接）下拉式列表中选择 `jdbc:derby://localhost:1527/sample`。IDE 将连接到数据库。

   如果 "Database Connection"（数据库连接）下未列出任何数据库，请等待片刻后再重试。Derby 是内建到 JDK 6 及更高版本中的数据库。

   如果您想要新建数据库连接，请参见___。
5. 从 "Database Table"（数据库表）下拉式列表中选择项目的主表。
6. 使用 "Available Columns"（可用列）和 "Columns to Include"（要包含的列）列表之间的方向键选择要在项目中包括的表列。

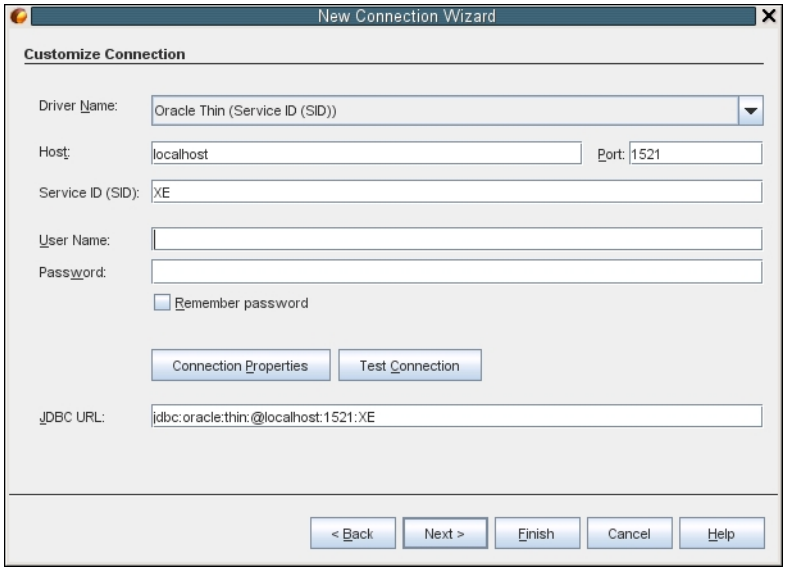


7. 单击 "Finish"（完成）。

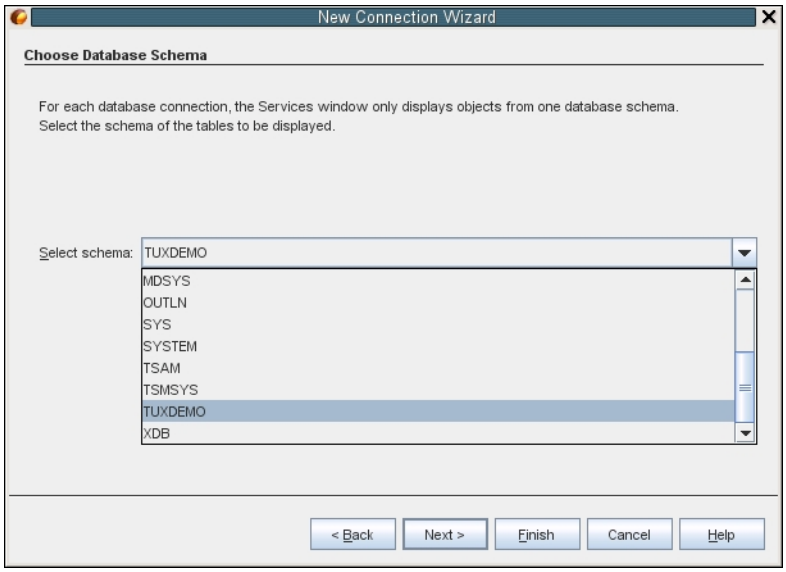
新建数据库连接

设置到 Oracle 数据库的新连接：

1. 在 "Master Table"（主表）页面上，选择 "New Database Connection"（新建数据库连接）创建新的数据库连接。



2. 在 "New Connection"（新建连接）向导的 "Customize Connection"（定制连接）页面上输入数据库地址以及您的用户名和口令，然后单击 "Next"（下一步）。
3. 在 "New Connection"（新建连接）向导的 "Choose Database Schema"（选择数据库方案）页面上，输入数据库方案或者从列表中进行选择。单击 "Finish"（完成）。



♦ ♦ ♦ 第 6 章

编辑和导航源文件

通过 IDE，您可以在编辑器中编辑和导航源文件。本章包含以下各节：

编辑源文件

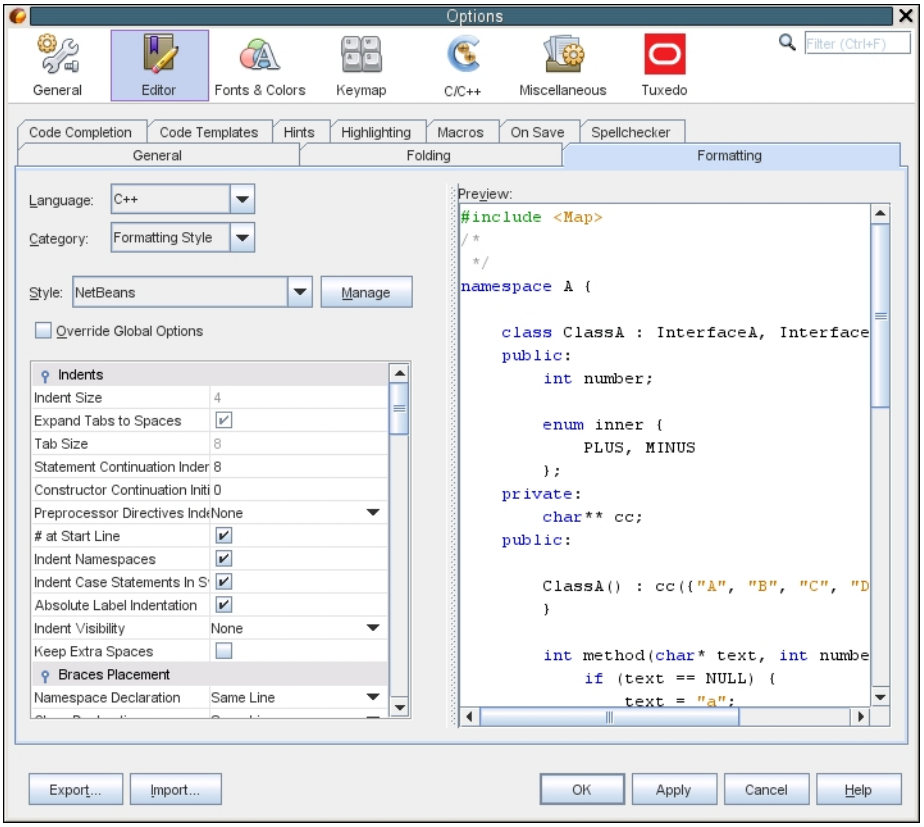
Oracle Solaris Studio IDE 提供了高级编辑和代码帮助功能，可帮助您查看和修改源代码。要了解这些功能，请使用 `Quote` 项目：

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

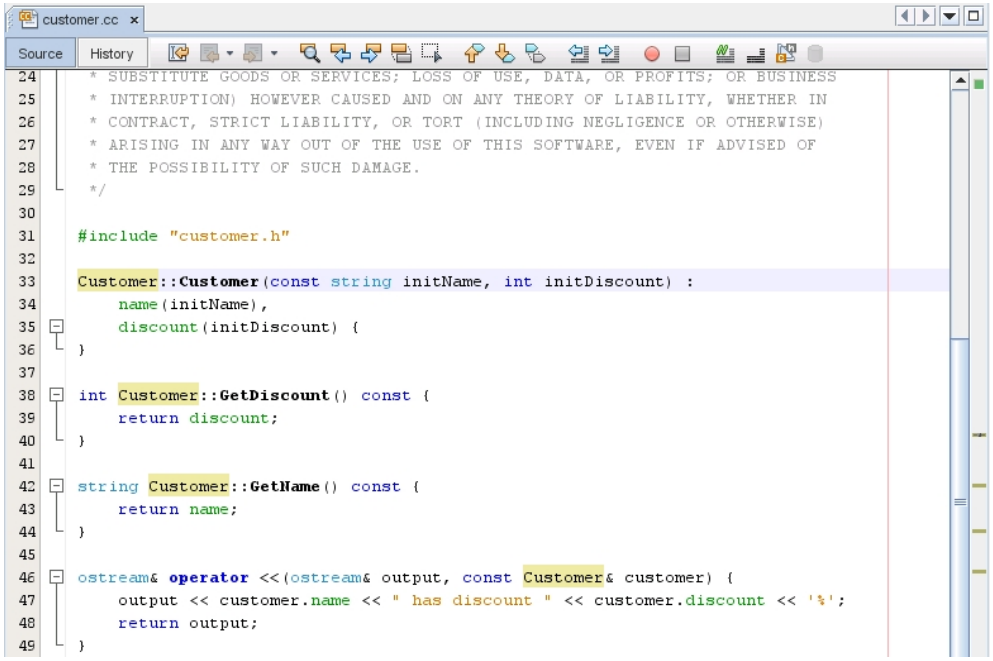
对于某些文件类型，您可以使用代码折叠功能折叠代码块，这样，只有块的第一行显示在源代码编辑器中。

1. 在 `Quote_1` 应用程序项目中，打开 "Source Files"（源文件）文件夹，然后双击 `cpu.cc` 文件以在源代码编辑器中将其打开。
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。

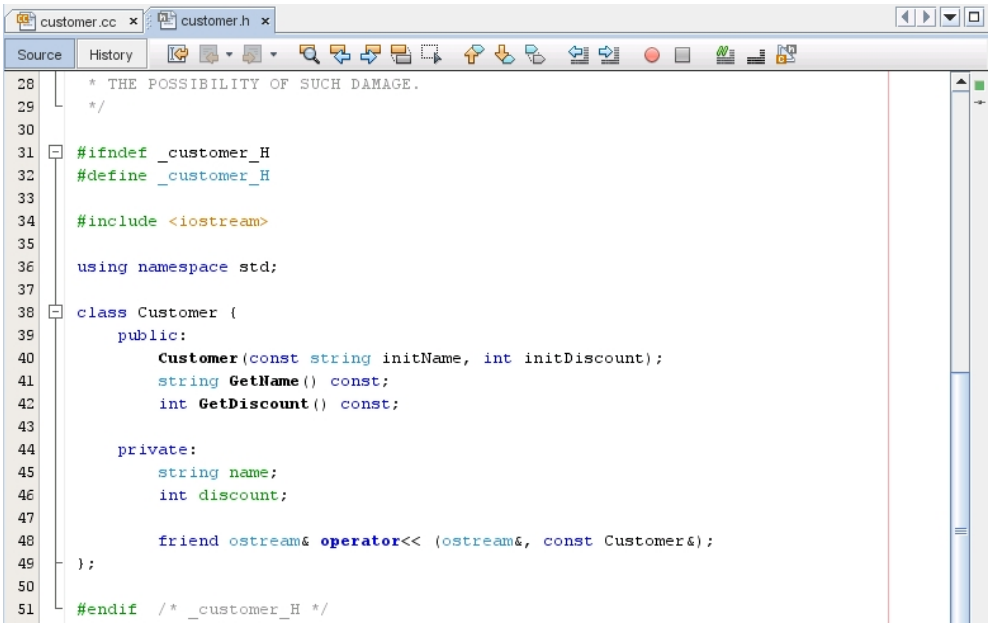
使用语义突出显示

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突出显示。

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"（突出显示）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，请注意函数名称是以粗体突出显示的。
7. 单击 `Customer` 类的某个实例。
8. 文件中 `Customer` 类的所有实例都会以黄色背景突出显示。



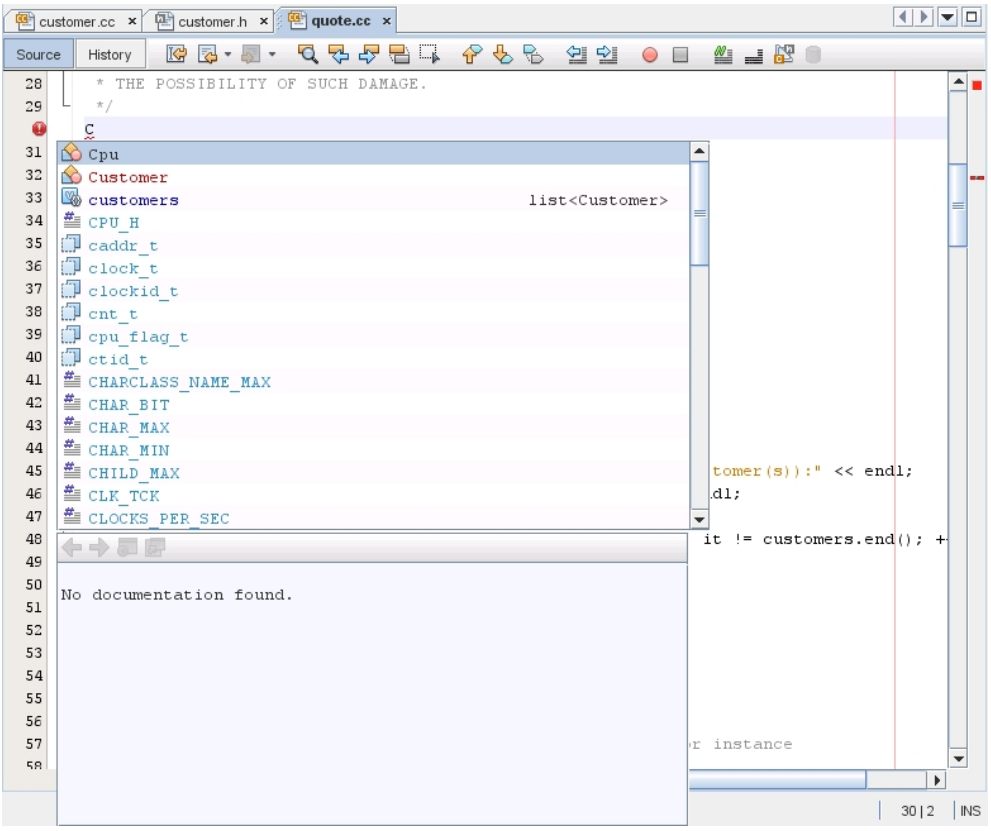
9. 在 `customer.h` 文件中，请注意类字段以粗体突出显示。



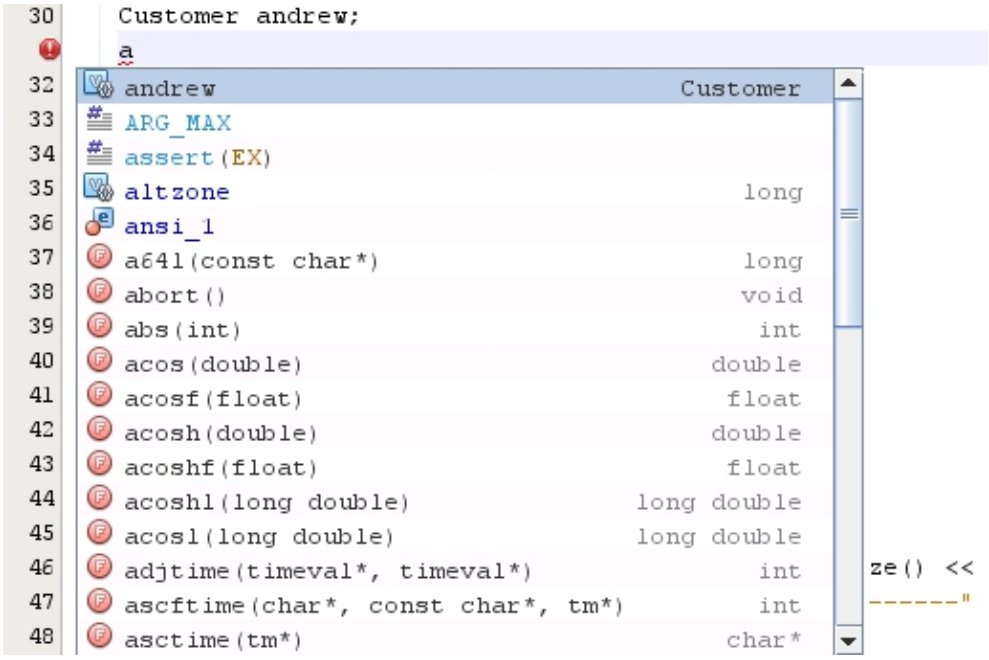
使用代码完成

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。

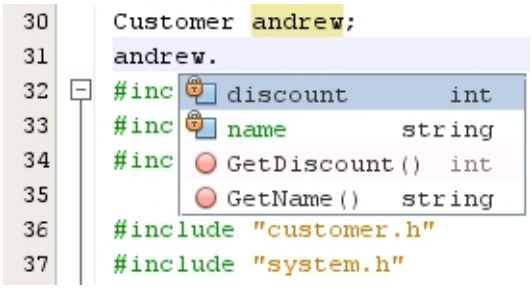
1. 打开 `Quote_1` 项目中的 `quote.cc` 文件。
2. 在 `quote.cc` 文件的第一个空白行上，键入大写的 C，然后按 Ctrl + 空格键。代码完成框将显示一个包含 `Cpu` 类和 `Customer` 类的简短列表。还会打开一个文档窗口，并显示消息 "No documentation found because the project source code does not include documentation"（找不到文档，因为项目源代码不包含文档）。
3. 通过再次按 Ctrl + 空格键，展开代码完成列表。



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 `andrew`。如果再次按 Ctrl + 空格键，代码完成框将显示一个包含以字母 a 开头、可从当前上下文访问的选项（如方法参数、类字段和全局名称）的列表。



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。

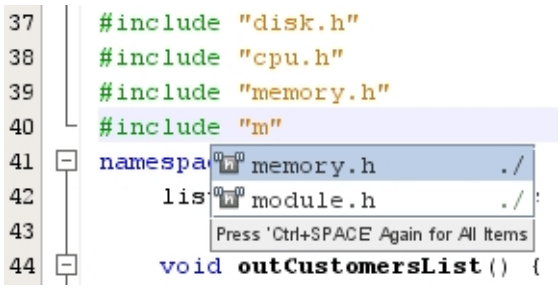


8. 删除所添加的代码。

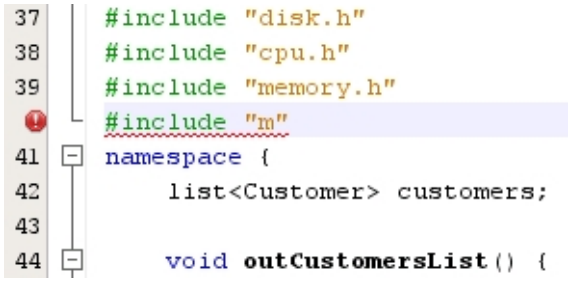
使用静态代码错误检查

当您在源代码编辑器中的源文件或头文件中键入代码时，编辑器会在您键入时执行静态代码错误检查，如果它检测到错误，则会在左边界处显示一个错误图标 。

1. 在 Quote_1 项目的 quote.cc 文件中，在第 40 行上键入 #include "m"。会弹出代码完成框，建议以 m 开头的两个头文件。



2. 在源代码编辑器中远离所添加代码的位置单击。请注意边界处显示的错误图标。



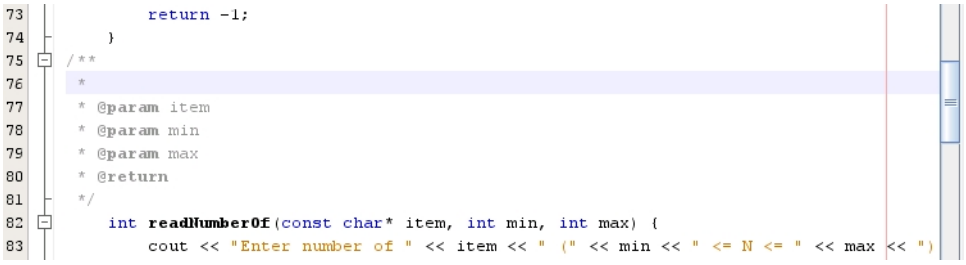
3. 退格至第二个引号，然后键入 odule.h" 来完成此语句，您会看到错误图标在语句引用现有的头文件后立即消失。
4. 删除所添加的语句。

有关如何选择要查看哪些错误或者禁用静态代码错误检查的更多信息，请参见 IDE 中的相关帮助页。

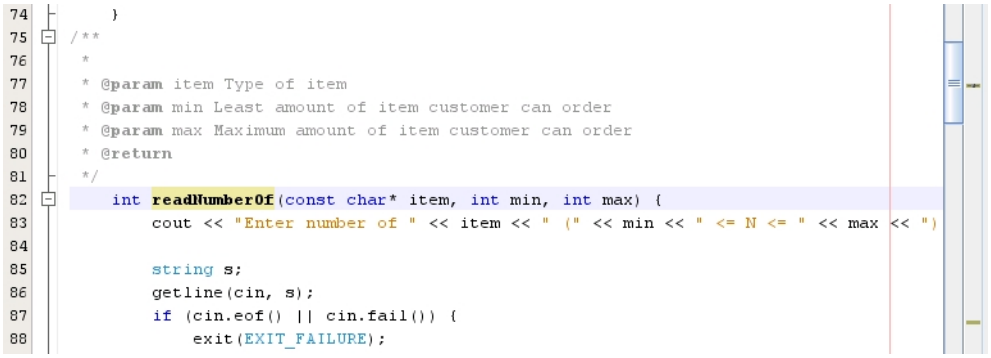
添加源代码文档

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。

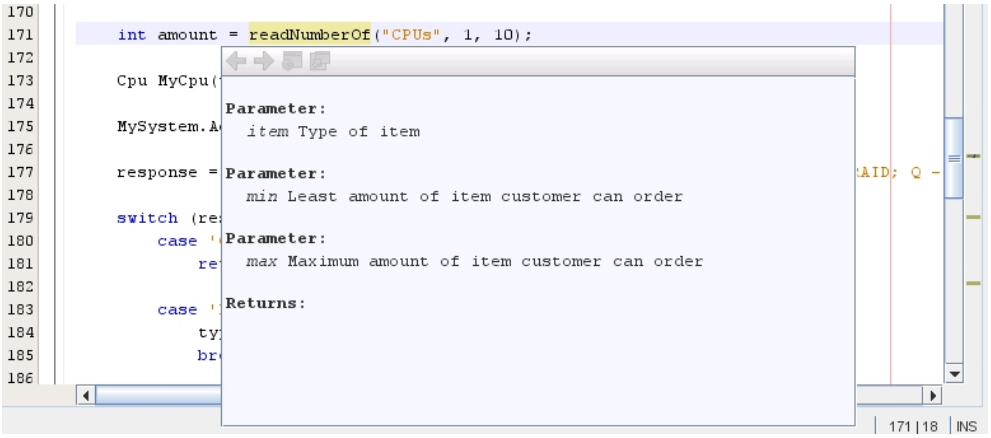
1. 在 quote.cc 文件中，将光标置于行 int readNumberOf(const char* item, int min, int max) { 上面的行上。
2. 键入一个正斜杠和两个星号 (/**)，然后按 Enter 键。编辑器会为 readNumberOf 类插入一条 Doxygen 格式的注释。



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。



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）再次为该类打开文档窗口。

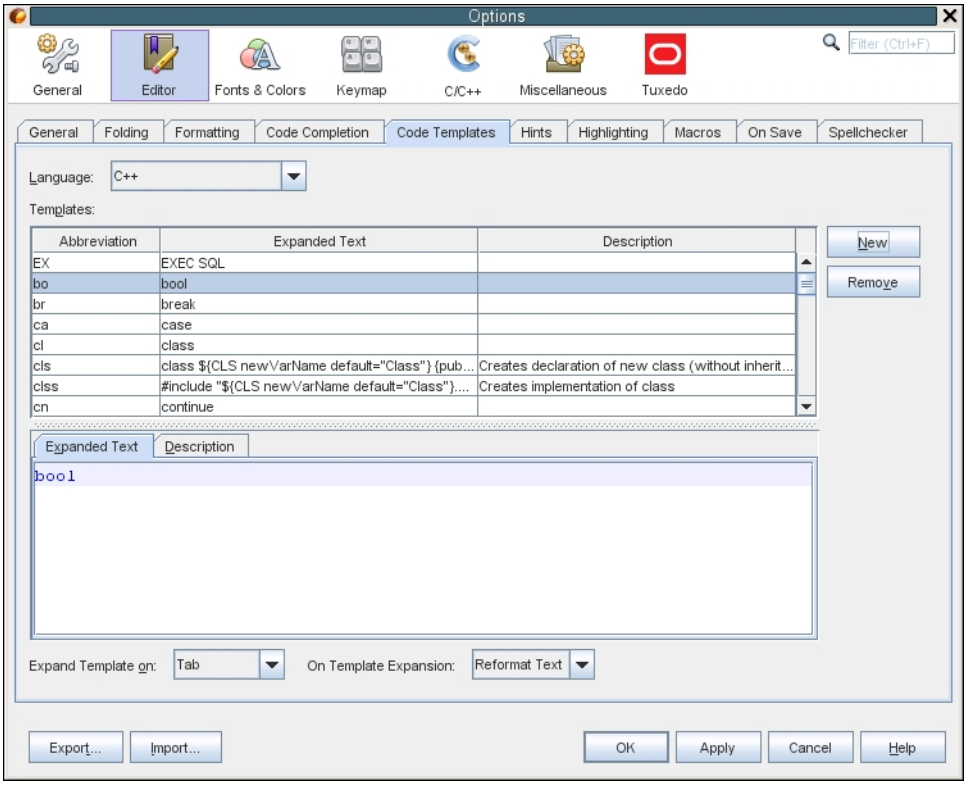
使用代码模板

源代码编辑器有一组用于 C、C++ 和 Fortran 代码的常见代码片段的可定制代码模板。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。
4. 使用 "New"（新建）或 "Remove"（删除）按钮添加或删除项目。您还可以在 "Expanded Text"（展开的文本）选项卡中编辑当前模板，或者在 "Description"（描述）选项卡中添加模板描述。



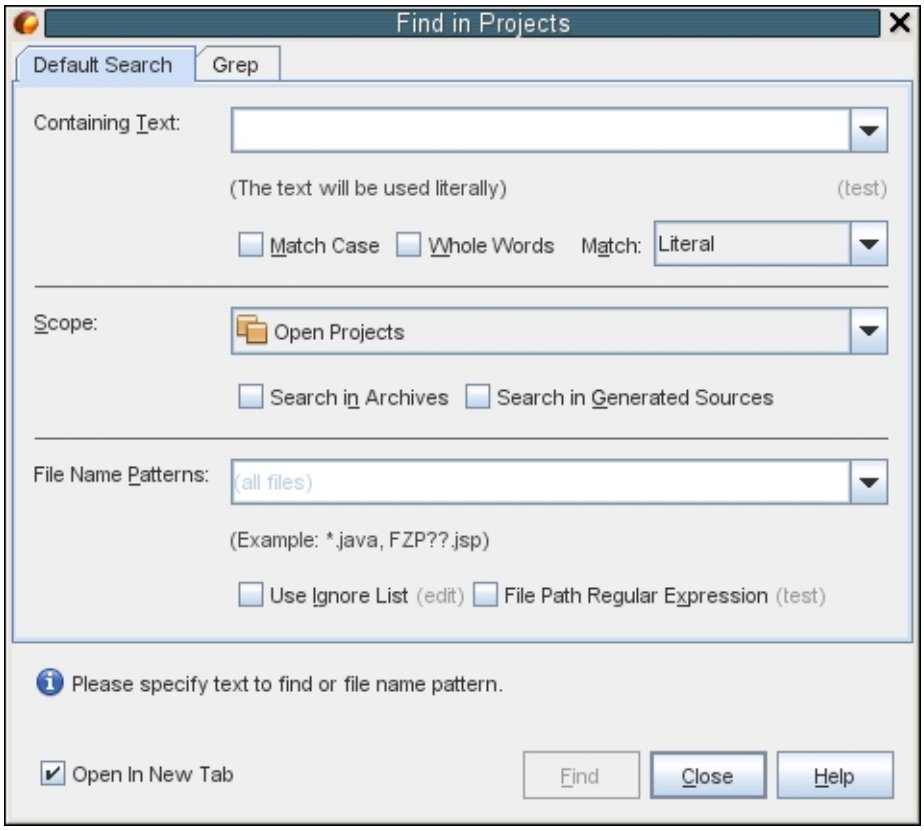
使用配对完成

当您编辑 C 和 C++ 源文件时，源代码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，源代码编辑器会自动插入封闭字符。

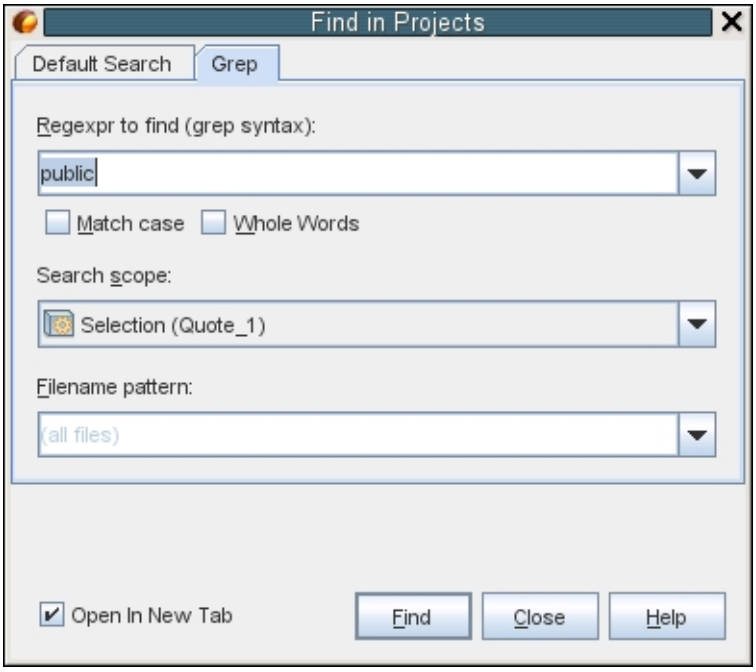
1. 在 `Quote_1` 项目中，将光标放到 `module.cc` 文件第 116 行上的 `{` 后面，然后按回车键打开一个新行。
2. 键入 `enum state {`，然后按 Return 键，将自动添加封闭花括号和分号，且光标位于两个花括号之间。
3. 键入 `invalid=0, success=1` 来完成枚举。
4. 在该行上枚举的封闭 `};` 后面，键入 `if (`，将自动添加封闭圆括号，且光标位于两个圆括号之间。
5. 键入 `v==null`。在右圆括号后键入 `i` 并换行，将自动添加封闭括号。
6. 删除所添加的代码。

在项目文件中查找文本

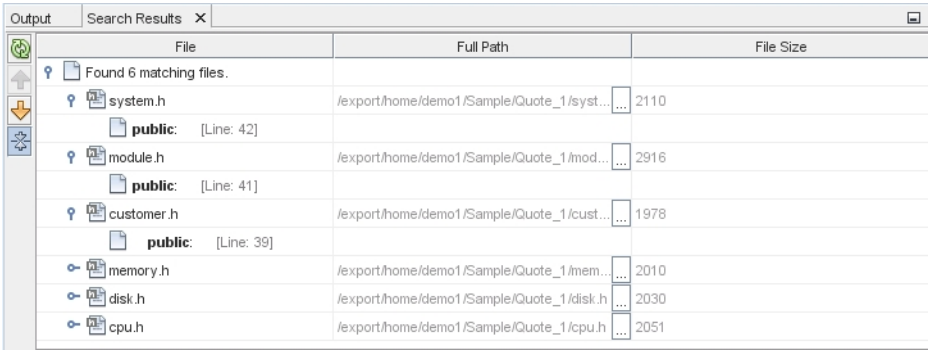
可以使用 "Find in Projects"（在项目中查找）对话框在项目中搜索指定文本或正则表达式的实例。

1. 通过以下方法之一打开 "Find in Projects"（在项目中查找）对话框：在 "Projects"（项目）窗口中右键单击项目并选择 "Find"（查找）；选择 "Edit"（编辑）> "Find In Projects"（在项目中查找）(Ctrl+Shift+F)。
2. 在 "Find in Projects"（在项目中查找）对话框中，选择 "Default Search"（缺省搜索）选项卡或 "Grep" 选项卡。"Grep" 选项卡使用 `grep` 实用程序，该实用程序提供较快的搜索，尤其适用于远程项目。
3. 

4. 在 "Grep" 选项卡中，键入要搜索的文本或正则表达式，指定搜索范围和文件名模式，然后选中 "Open in New Tab"（在新选项卡中打开）复选框，这样您就可以在单独的选项卡中保存多个搜索了。



5. 单击 "Find"（查找）。



"Search Results"（搜索结果）选项卡会列出已在其中找到相应文本或正则表达式的文件。

使用左边界中的按钮可以更改搜索结果的视图。

6. 单击 "Expand/Collapse"（展开/折叠）按钮可折叠文件列表，以便只显示文件名。
7. 双击列表中的某一项，IDE 将转到源代码编辑器中的对应位置。

查找和替换

您可以使用编辑器中的 "Find/Replace"（查找/替换）功能，它作为一个整体位于编辑器窗口底部的 "Find"（查找）工具栏中，而不是一个单独的用于替换的对话框。"Replace"（替换）字段和按钮显示在 "Find"（查找）字段和按钮下的工具栏中。按 Ctrl+F 可激活 "Find"（查找）工具栏，按 Ctrl+H 可激活 "Replace"（替换）功能。

使用剪贴板历史记录

找到要查找的文本后，可能想要复制此文本并在代码的其他区域中使用它。

您可以查看复制到桌面剪贴板的最后九个文本缓冲区，并选择一个进行粘贴。将光标放在想要插入文本的位置，按 Ctrl+Shift+D 可打开包含剪贴板条目的弹出窗口。使用箭头键可导航剪贴板缓冲区，并在缓冲区列表下的窗口中查看全部内容。要粘贴缓冲区的内容，请键入缓冲区编号或者在选择缓冲区后按 Enter 键。请注意，此缓冲区包含从桌面上任意窗口复制过来的内容，而不仅仅是 IDE 的内容。

提示 - 您可以通过将鼠标光标悬停在 IDE 中的任意文件上并按 Alt+Shift+L 将文件路径复制到剪贴板中。

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。

窗口管理和分组

除了单个窗口外，您还可以对窗口组执行操作。对于每个窗口所属的组，您可以将其最小化、拖动到新位置、浮动在单独的窗口中或者停靠回 IDE 窗口。

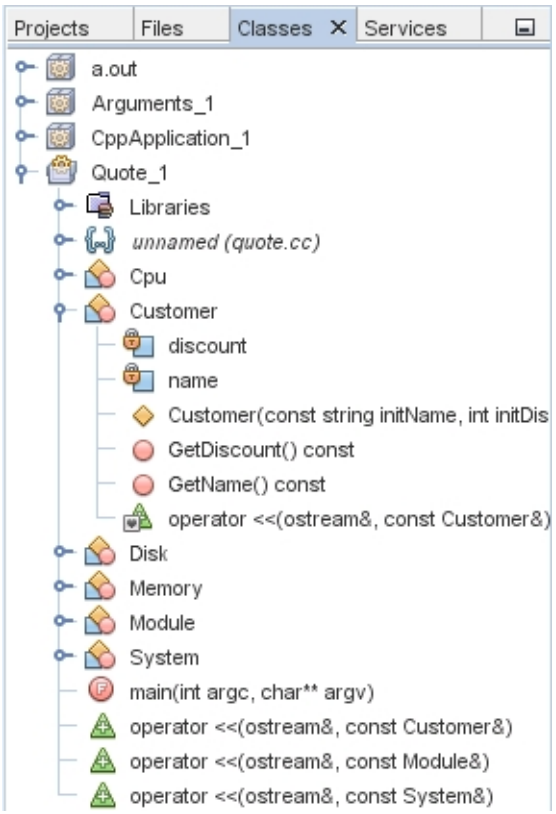
1. 通过单击组右侧的 "Minimize Window Group"（最小化窗口组）按钮，可以最小化左上部的 "Projects"（项目）、"Files"（文件）、"Classes"（类）和 "Services"（服务）窗口。
2. 在组的选项卡区域内单击右键或者选择 "Window"（窗口）> "Configure Windows"（配置窗口）并选择 "Maximize"（最大化）(Shift+Escape)，可以最大化窗口组。

   请注意，您可以为窗口组选择其他选项，例如 "Float"（浮动）或者"Dock"（停靠）。

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。



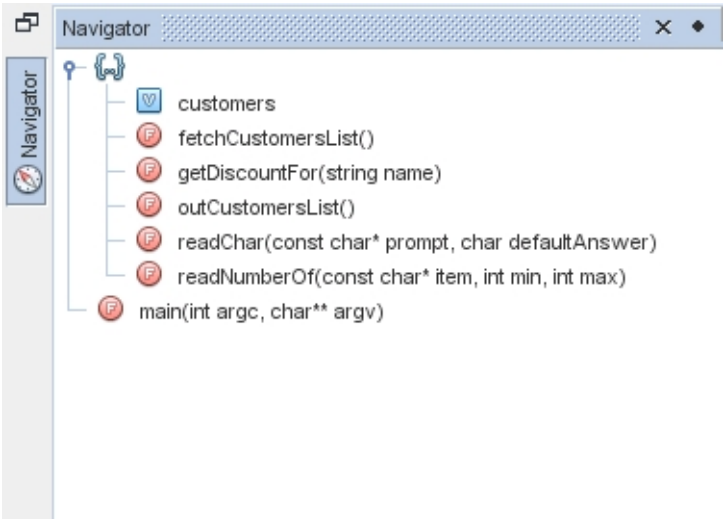
4. 双击 `name` 变量可打开 `customer.h` 头文件。

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）(Ctrl-7) 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。

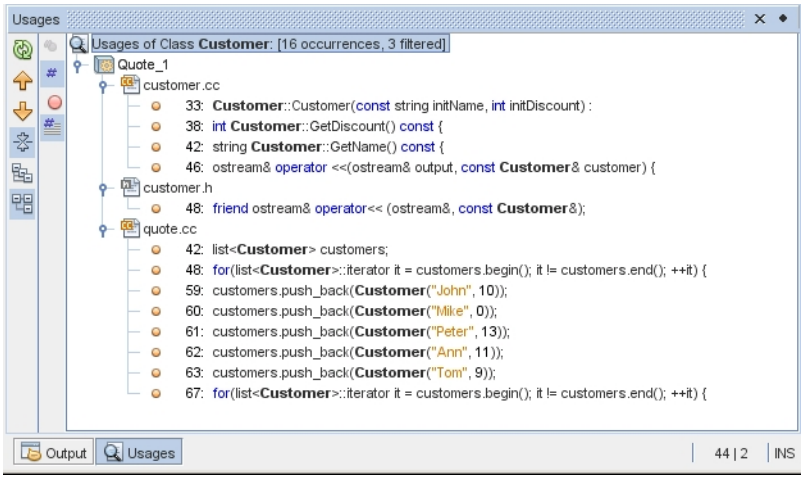


3. 要导航到文件的某个元素，请在 "Navigator"（导航器）窗口中双击该元素，"Editor"（编辑器）窗口中的光标将移到该元素。
4. 在 "Navigator"（导航器）窗口中右键单击，可看到用于在窗口中对元素进行排序、对项目进行分组或过滤的选项。
5. 要查看 "Navigator"（导航器）窗口的图标所表示的意义，请通过选择 "Help"（帮助）> "Help Contents"（帮助内容）打开 IDE 联机帮助。在 "Help"（帮助）浏览器中，单击 "Search"（搜索）选项卡，然后在 "Find"（查找）字段中键入 `navigator icons`（导航器图标）。

查找类、方法和字段使用实例

您可以使用 "Usages"（使用实例）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，选择 "Find Usages"（查找使用实例）(Alt-F7)。
2. 在 "Find Usages"（查找使用实例）对话框中，单击 "Find"（查找）。
3. "Usages"（使用实例）窗口将打开，并显示项目的源文件中 `Customer` 类的所有使用实例。

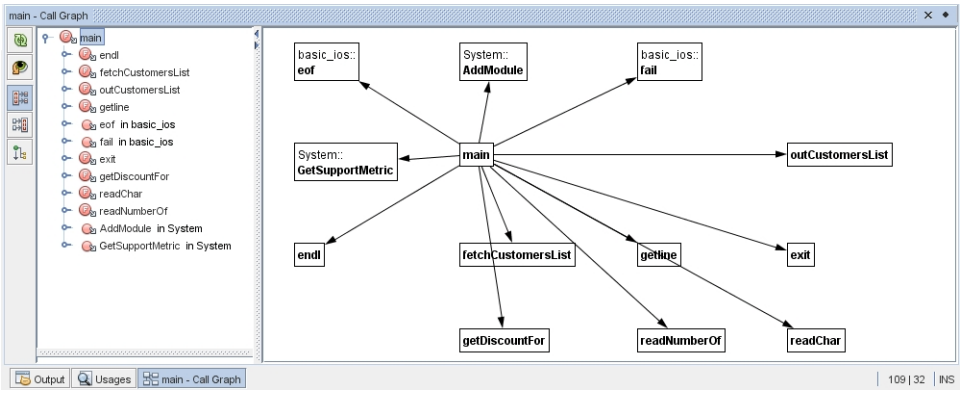


"Find Usages"（查找使用实例）在后台运行，这样在搜索大量文件时您还可以执行其他任务。"Usages"（使用实例）窗口会在以增量方式找到新结果后进行更新。将会显示进度指示器以及递增的实例数。您可以随时停止搜索，系统将保存截止至该时间点的搜索结果。您可以在搜索命中内容之间进行导航，将视图从逻辑更改为物理，并使用不同的设置再次运行 "Find Usages"（查找使用实例）。

使用调用图

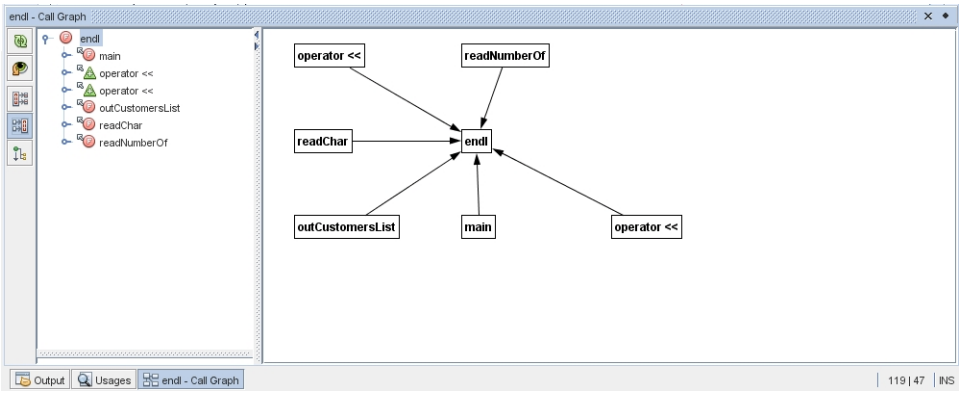
"Call Graph"（调用图）窗口显示类中函数之间的调用关系的两个视图。树视图显示从某个选定函数调用的函数，或者调用该选定函数的函数。图形视图使用箭头来显示被调用函数与调用函数之间的调用关系。

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）。
2. "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。

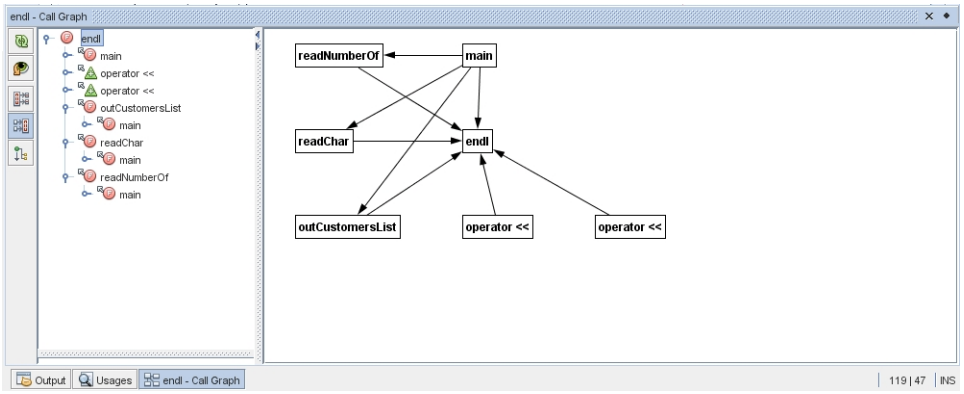


如果屏幕抓图中没有显示所有函数，请单击 "Call Graph"（调用图）窗口左侧的 "Who is Called From the Function"（此函数中调用的函数）按钮以便显示从 main 函数调用了哪个函数。

3. 展开 `end1` 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 `end1` 所调用的函数。
4. 选择 `end1` 节点，单击窗口左侧的 "Bring Into Focus"（聚焦）按钮以将焦点放到 `endl` 函数上，然后单击 "Who Calls This Function"（调用此函数的函数）按钮来查看调用 `end1` 函数的所有函数。



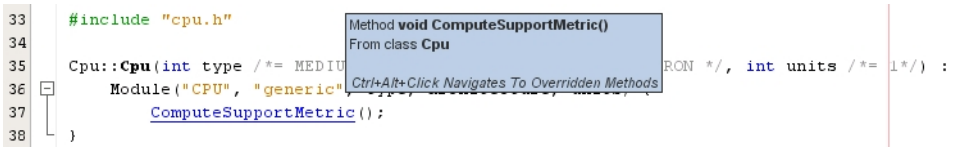
5. 展开树中的某些节点可看到更多函数。



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。



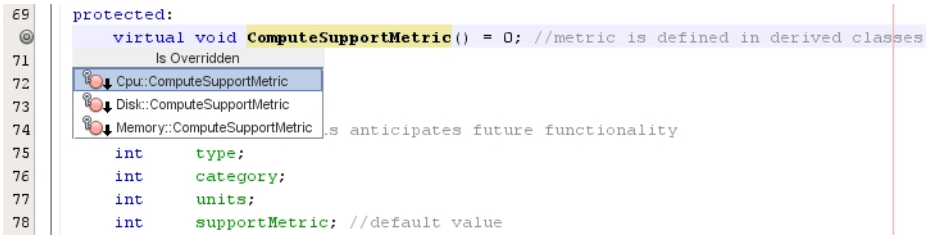
2. 单击超级链接，编辑器将跳到函数的定义。

```
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
46  void Cpu::ComputeSupportMetric() {
47      int metric = 100 * GetUnits();
48
```

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。编辑器会跳转至 `cpu.h` 头文件中的函数声明。
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的绿色圆圈 上，此时会显示注释，指示此方法会覆盖其他方法。

```
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();
48
49      switch (GetTypeID()) {
50          case MEDIUM:
```

6. 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。
7. 单击灰色圆圈，编辑器会显示覆盖此方法的方法列表。

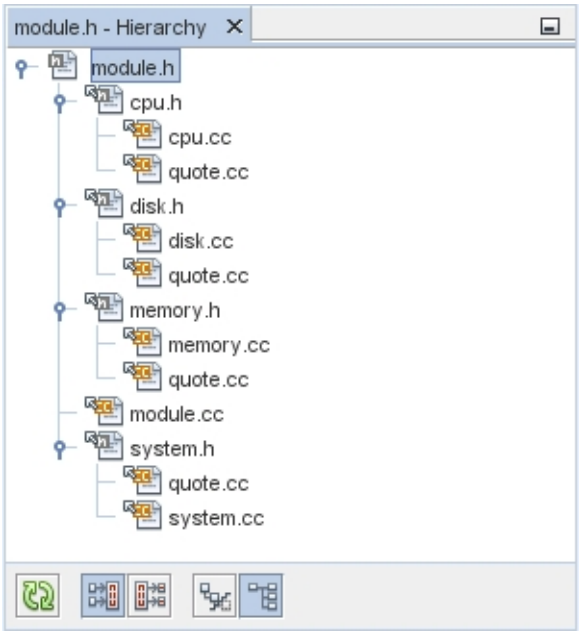


8. 单击 `Cpu::ComputerSupportMetric` 项，编辑器会跳回到 `cpu.h` 头文件中该方法的声明。

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。

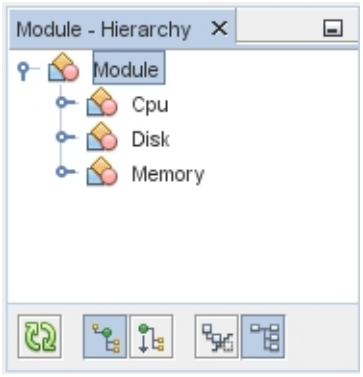
1. 在 `Quote_1` 项目中，打开源代码编辑器中的 `module.cc` 文件。
2. 右键单击文件中的 `#include "module.h"` 行，然后选择 "Navigate"（导航）> "View Includes Hierarchy"（查看包含分层结构）。
3. 缺省情况下，"Hierarchy"（分层结构）窗口显示直接包含头文件的文件的普通列表。单击窗口底部的 "Show Tree View"（显示树视图）按钮。单击 "Show Direct Includes Only"（仅显示直接包含）按钮以显示包含或被包含的所有文件。展开树视图中的节点可看到包含头文件的所有源文件。



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。


Module - Hierarchy
Module
Cpu
Disk
Memory

♦ ♦ ♦ 第 7 章

7 调试应用程序

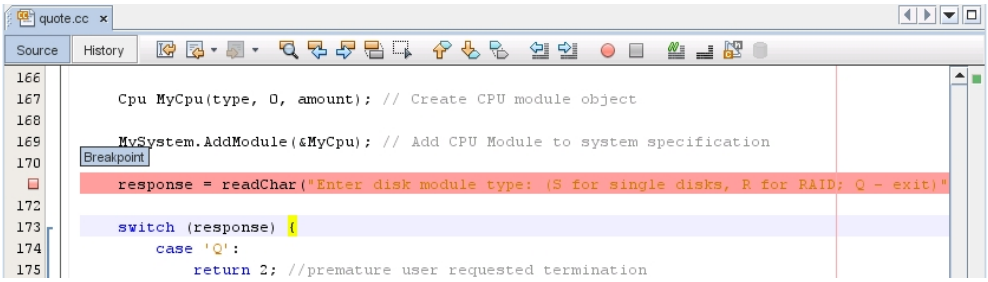
本章讨论了如何为项目创建断点以及如何使用这些断点来调试代码。本章包含以下各节：

创建断点

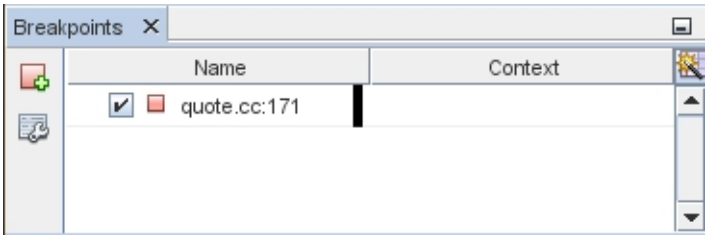
您可以随时创建和处理代码中的断点。为了运行调试器，您必须在代码中设置断点以便调试器知道在哪里暂停执行，从而使您可以检查变量值，逐行执行代码行，并修复错误。

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 171 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。

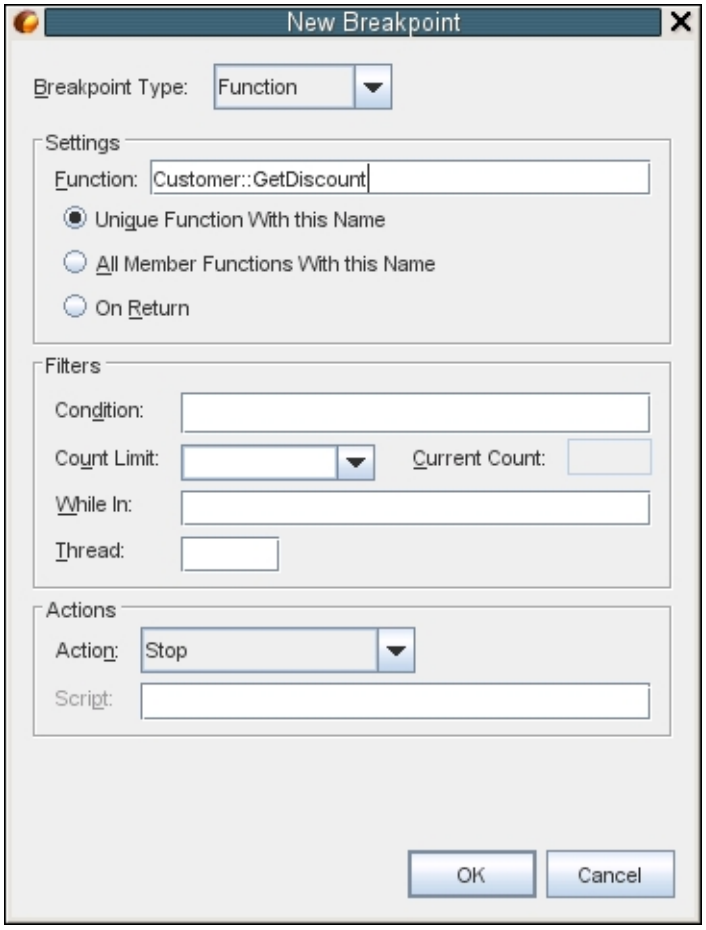


3. 您可以通过单击左边界中的图标移除断点。
4. 选择 "Window"（窗口）> "Debugging"（调试）> "Breakpoints"（断点）可打开 "Breakpoints"（断点）窗口。该窗口中会列出您的行断点。

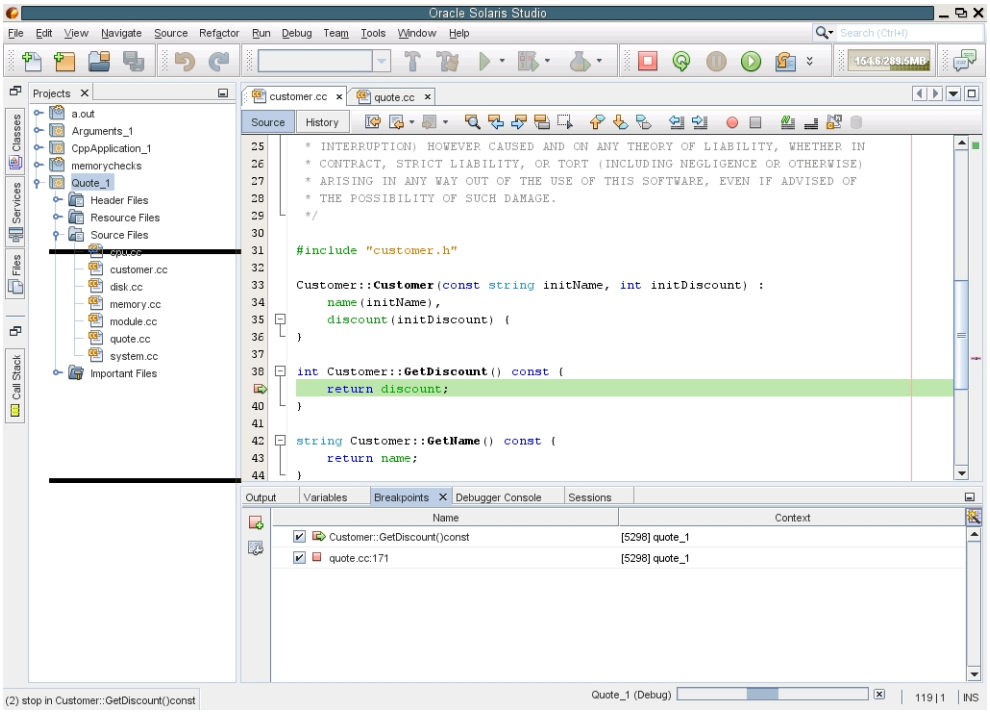


创建函数断点

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。



将断点分组

您可以使用多个不同类别将断点分组，例如按文件、按项目、按类型、按语言等。

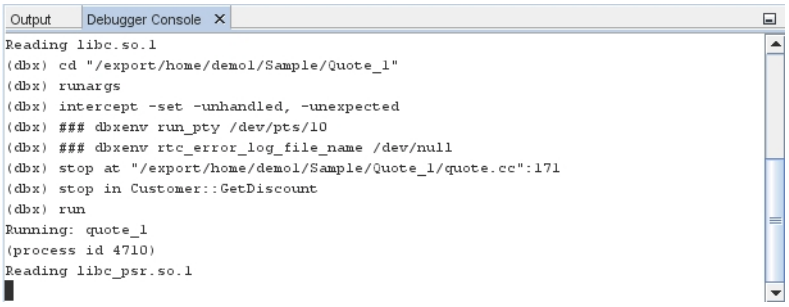
1. 选择 "Window"（窗口）> "Debugging"（调试）> "Breakpoints"（断点）窗口。
2. 单击 "Select Breakpoint Groups"（选择断点组）图标。
3. 选择断点组。
   断点将根据您的选择进行排列。

调试项目

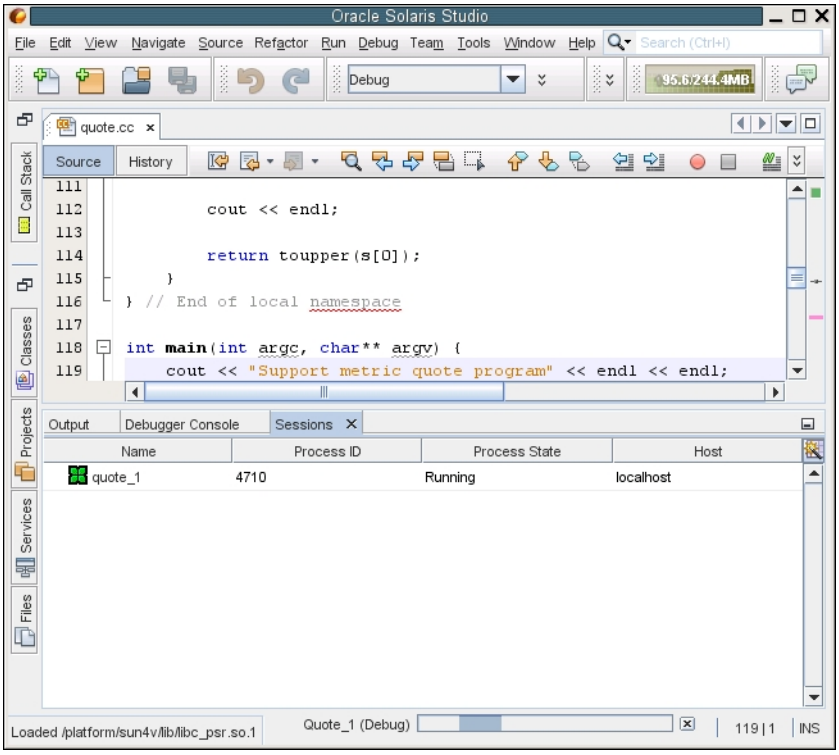
当您启动调试会话时，IDE 将启动项目的关联工具集中的调试器（缺省情况下为 dbx 调试器），然后在该调试器内运行应用程序。IDE 会自动打开调试器窗口，并将调试器输出内容输出到 "Debugger Console"（调试器控制台）窗口。

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，同时 "Debugger Console"（调试器控制台）窗口将打开。

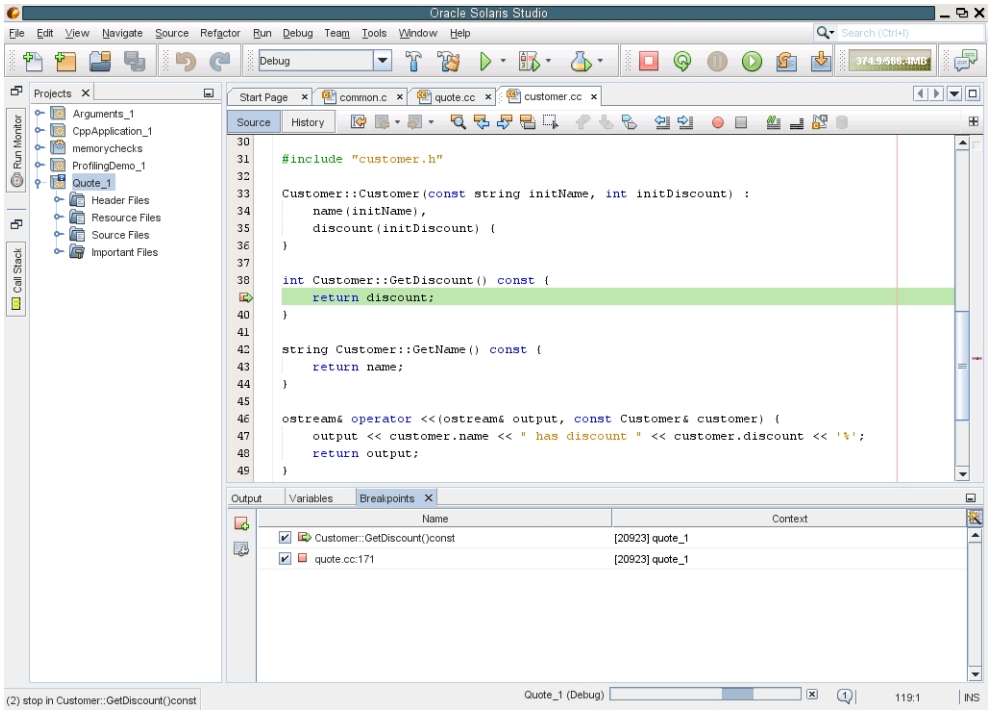


2. 通过选择 "Window"（窗口）> "Debugging"（调试）> "Variables"（变量） (Alt +Shift-1) 打开 "Variables"（变量）窗口。
3. 选择 "Window"（窗口）> "Debugging"（调试）> "Sessions"（会话）(Alt+Shift-6) 打开 "Sessions"（会话）窗口。调试会话将显示在此窗口中。

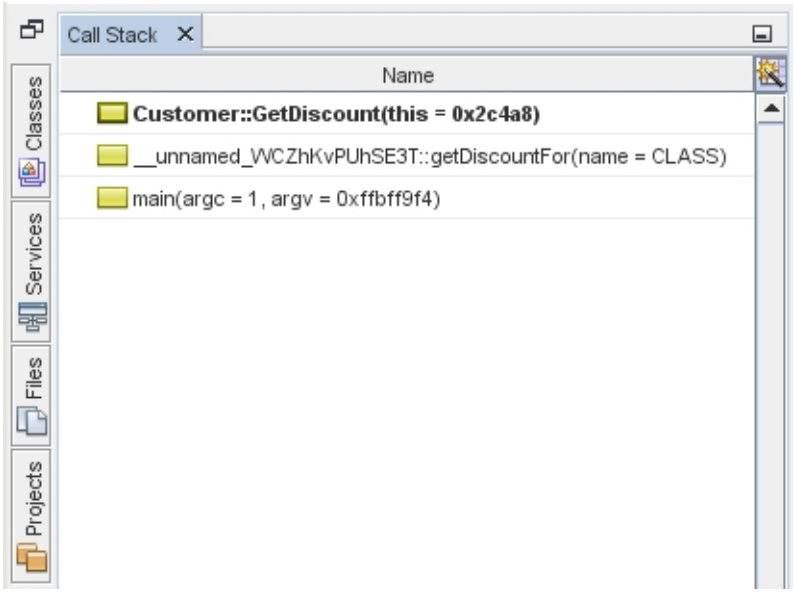


检查应用程序的状态

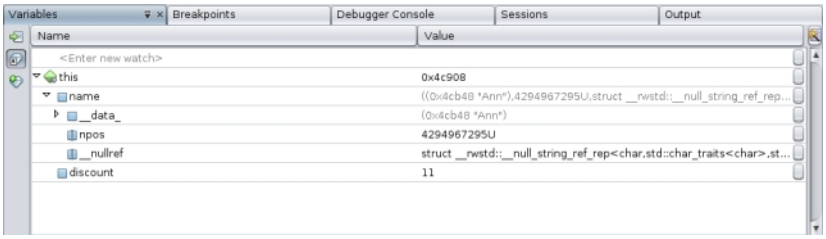
1. `Quote_1` 应用程序在 "Output"（输出）窗口中提示您进行输入。在 `Enter customer name:` 提示符后输入客户名称。
2. 在 `customer.cc` 文件中，绿色的程序计数器箭头显示在 `GetDiscount` 函数第一行上的断点图标顶部。



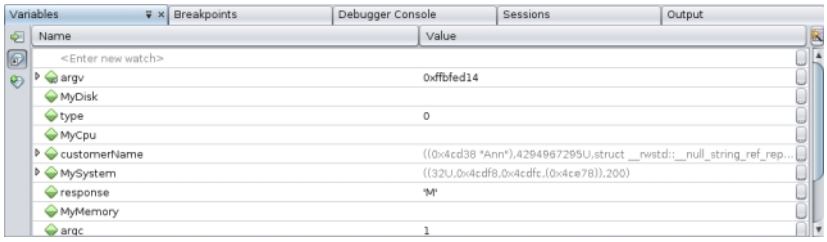
3. 打开 "Call Stack"（调用堆栈）窗口 (Alt+Shift-3)。调用堆栈将显示三个帧。



4. 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。



5. 单击 "Continue"（继续）按钮。`GetDiscount` 函数将执行，它将客户折扣输出到 "Output"（输出）窗口中。然后系统会提示您进行输入。
6. 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。



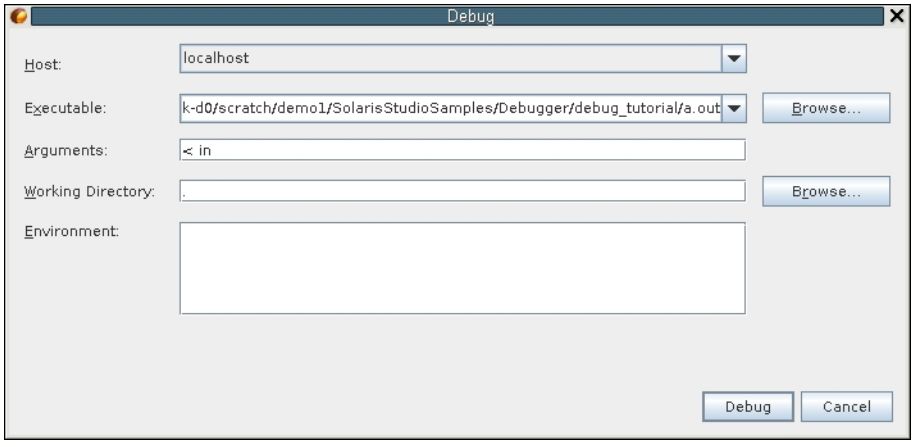
7. 查看 "Call Stack"（调用堆栈）窗口，您会看到，堆栈中只有一个帧。
8. 单击 "Continue"（继续） 并根据 "Output"（输出）窗口中的提示继续输入内容，直到程序完成。将最后的输入内容输入到程序中后，调试会话将结束。要在程序执行完成之前结束调试会话，可以在 "Sessions"（会话）窗口中右键单击该会话，然后选择 "Finish"（完成）。

调试可执行文件

您可以调试不在 IDE 项目中的可执行二进制文件。但是，应通过用于生成可执行文件的源代码来查找可执行文件，这样调试器便可以查找调试信息。这也称为“无项目调试”。

调试可执行文件：

1. 在菜单中，选择 "Debug"（调试）> "Debug Executable"（调试可执行文件）。



2. 在 "Debug"（调试）对话框中，从 "Host"（主机）下拉列表中选择要使用的开发主机。

3. 单击 "Browse"（浏览）按钮查找可执行文件路径，或者直接在 "Executable"（可执行文件）文本字段中键入路径。如果单击 "Browse"（浏览）查找可执行文件路径，会自动生成 "Working Directory"（工作目录）文本字段。
4. 在 "Arguments"（参数）文本字段中添加任意参数。
5. 如果未指定 "Working Directory"（工作目录），请直接在 "Working Directory"（工作目录）文本字段中键入工作目录路径；或者，请单击 "Browse"（浏览），在 "Working Directory"（工作目录）对话框中选择目录，然后单击 "Select"（选择）。
6. 通过在 "Environment"（环境）窗格中键入设置来指定所有环境变量。
7. 单击 "Debug"（调试）。

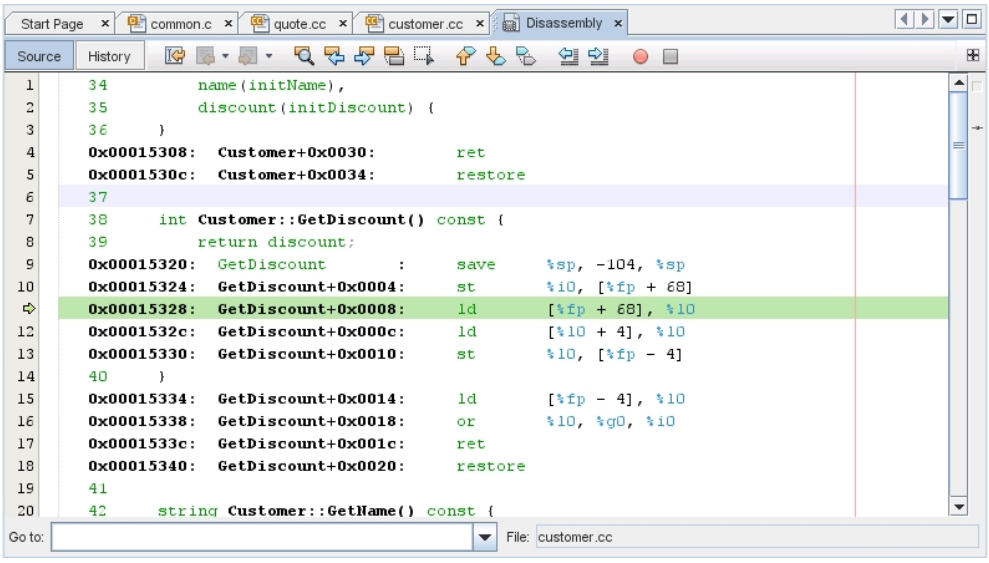
您选择的程序将装入 IDE，但是处于 "Pause"（暂停）状态。您可以单击 "Continue"（继续）来继续进行调试。

有关调试可执行文件的更多信息，请参见相应的 IDE 帮助页。

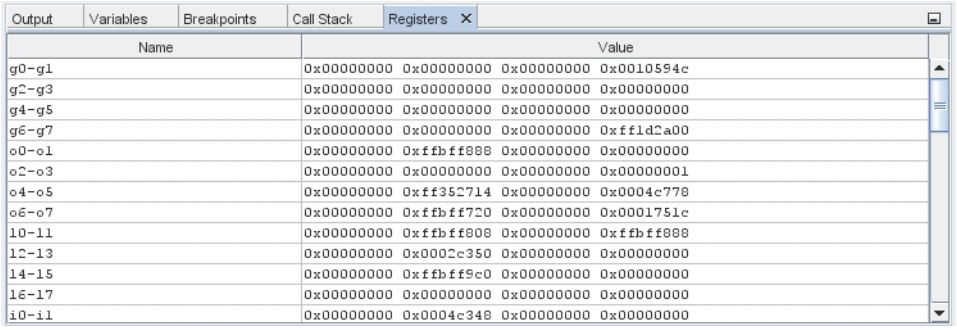
在机器指令级调试

调试器提供了用于在机器指令级调试项目的窗口。

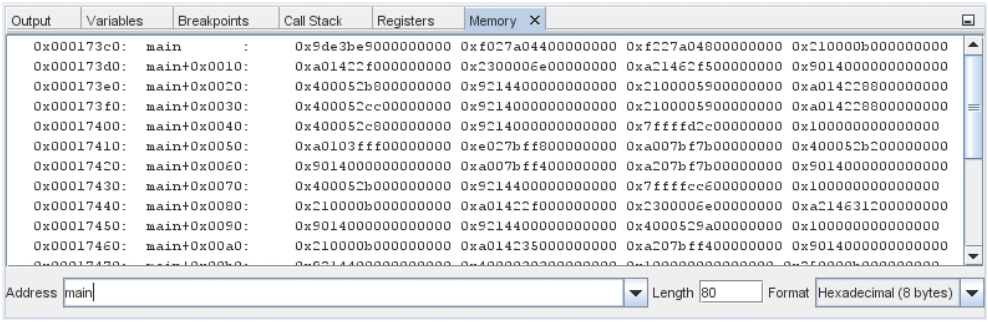
1. 右键单击 `Quote_1` 项目，然后选择 "Debug"（调试）。
2. 在 "Output"（输出）窗口中，根据提示输入客户名称。
3. 当程序在 `GetDiscount` 函数上的断点处暂停时，选择 "Window"（窗口）> "Debugging"（调试）> "Disassembly"（反汇编）可打开 "Disassembly"（反汇编）窗口，如同在 "Editor"（编辑器）窗口中一样。绿色程序计数器箭头将显示在暂停程序的指令处的断点图标顶部。



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。



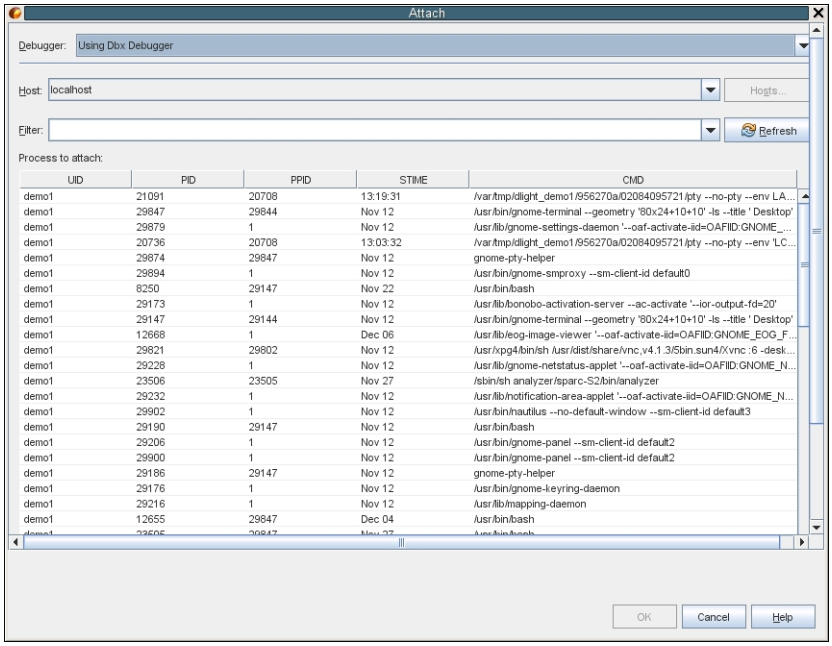
5. 选择 "Window"（窗口）> "Debugging"（调试）> "Memory"（内存）可打开 "Memory"（内存）窗口，该窗口显示项目当前使用内存的内容。在窗口底部，可以指定要浏览的内存地址、更改内存浏览的长度，或者更改内存信息的格式。



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）。项目将生成，`Freeway` 应用程序将启动。在 `Freeway` GUI 窗口中，选择 "Actions"（操作）> "Start"（启动）。
5. 在 IDE 中，选择 "Debug"（调试）> "Attach Debugger"（附加调试器）。



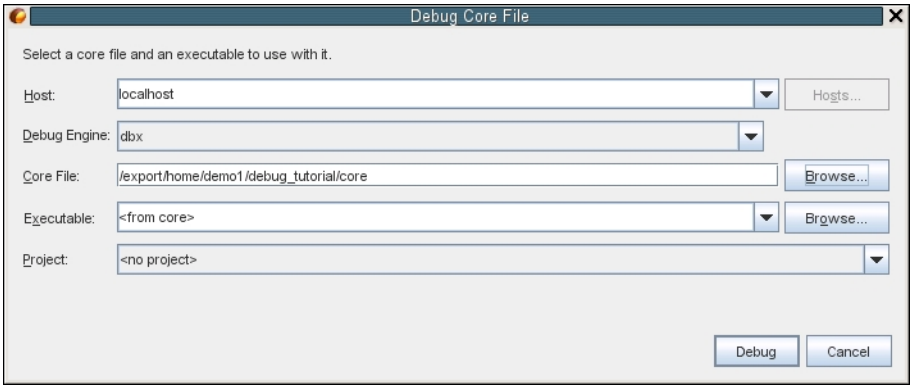
6. 在 "Attach"（附加）对话框中，在 "Filter"（过滤器）字段中键入 `Freeway` 以过滤处理器列表。
7. 从过滤后的列表中选择 `Freeway` 进程。
8. 单击 "OK"（确定）。
9. 将启动一个调试会话，`Freeway` 进程会在执行到调试器连接到该进程的位置暂停。
10. 单击 "Continue"（继续）  继续执行 `Freeway`，它当前在调试器控制下运行。如果单击 "Pause"（暂停）  ，将暂停执行 `Freeway`，然后您可以检查变量、调用堆栈，等等。
11. 再次单击 "Continue"（继续），然后单击 "Finish Debugger Session"（完成调试器会话）  。调试器会话将结束，但 `Freeway` 进程会继续执行。在 `Freeway` GUI 中选择 "File"（文件）> "Exit"（退出）以退出应用程序。

调试信息转储文件

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

1. 选择 "Debug"（调试）> "Debug core file"（调试信息转储文件）。

2. 在 "Debug core file"（调试信息转储文件）字段中键入信息转储文件的完整路径，或者在 "Select Core File"（选择信息转储文件）对话框中单击 "Browse"（浏览）并导航到您的信息转储文件。



3. 如果调试器无法将您指定的信息转储文件与某个可执行文件相关联，它将显示一条错误消息。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）按钮并使用 "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）。

有关调试的更深入的教程，请参见《Oracle Solaris Studio 12.4：dbxtool 教程》。

♦ ♦ ♦ 第 8 章

监视项目

IDE 提供了工具，用于检查正在运行的 C/C++/Fortran 项目的行为，以便您检测应用程序中的运行时问题。这些类型的问题无法在调试代码时检测到。分析工具包括：

- CPU 使用情况
- 线程使用情况
- 内存使用情况
- 内存访问错误

本章介绍这些工具以及如何使用它们，包含以下各节：

使用“运行监视器”分析工具来分析项目

“运行监视器”分析信息使用 Oracle Solaris Studio 性能分析工具以及底层操作系统进行收集。“运行监视器”工具以图形方式显示信息，其中包含一些按钮，单击这些按钮可以显示有关代码中问题区域的更多信息。本节将设置一个分析演示项目，在项目的属性中启用分析工具并检查“运行监视器”分析工具的结果。

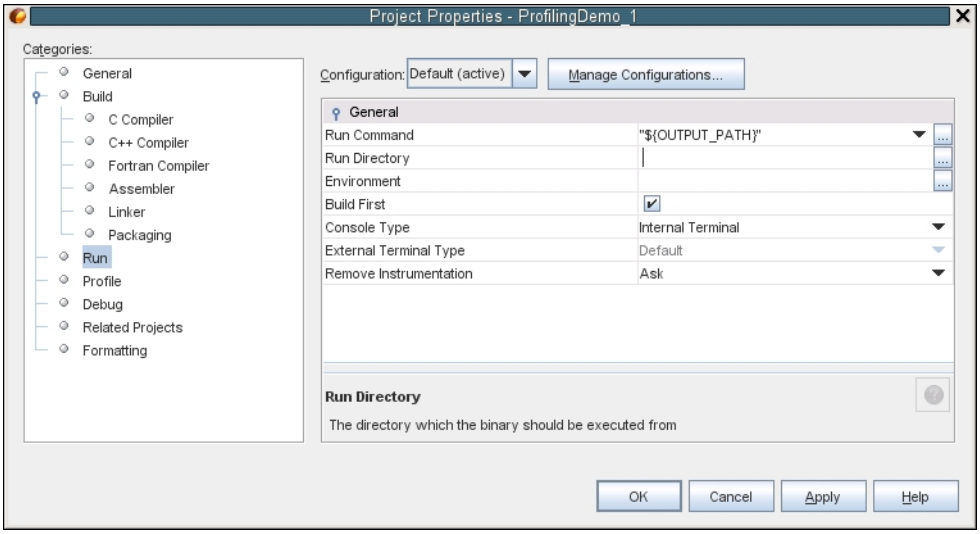
设置分析项目

本教程将使用 ProfilingDemo 样例项目。创建此演示：

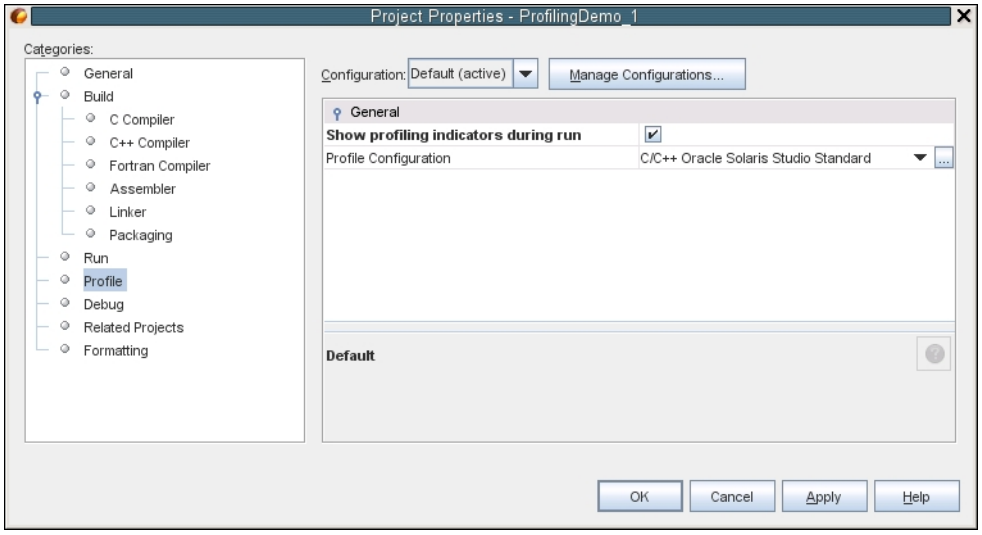
1. 选择 "File"（文件）> "New Project"（新建项目）(Ctrl+Shift+N)。
2. 在 "New Project"（新建项目）向导中，展开 "Samples"（样例）节点，然后选择 "C/C++" 类别。
3. 选择分析演示项目。单击 "Next"（下一步），然后单击 "Finish"（完成）。

配置项目并启用分析：

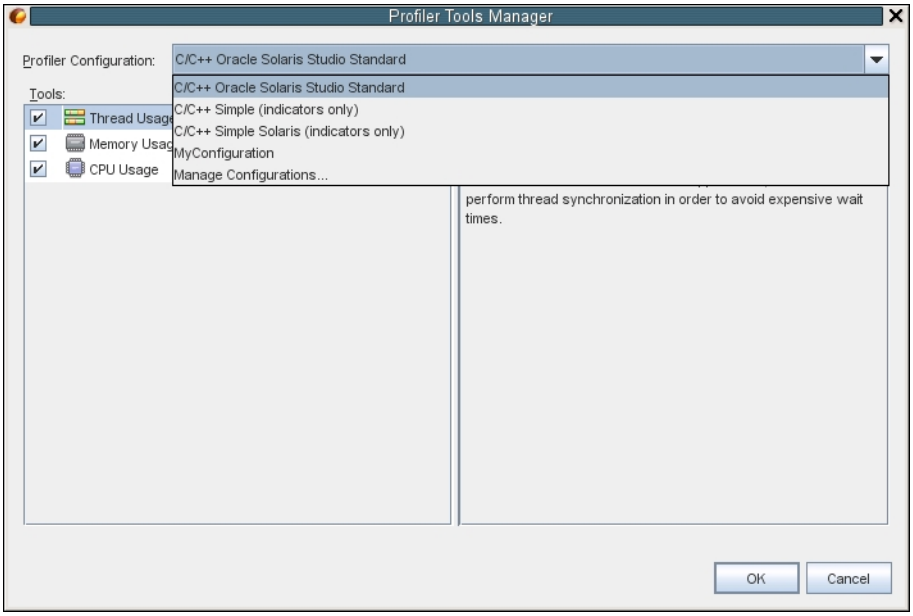
1. 右键单击 "Projects"（项目）选项卡中的 "ProfilingDemo_1" 项目节点，然后选择 "Properties"（属性）。
2. 选择 "Categories"（类别）面板中的 "Build"（生成）节点，并确保 "Tool Collection"（工具集合）设置为 "Oracle Solaris Studio "。
3. 选择 "Categories"（类别）面板中的 "Run"（运行）节点，并确保 "Console Type"（控制台类型）设置为 "Internal Terminal"（内部终端）。这样，就可以在 IDE 的 "Output"（输出）窗口而不是外部终端窗口中查看程序输出。



4. 选择 "Categories"（类别）面板中的 "Profile"（分析）节点。注意选中了 "Show profiling indicators during run"（在运行过程中显示分析指示器）选项。可以在任何项目中选中此选项以显示 "Run Monitor Tools"（运行监视器工具）选项卡。



5. 在 "Profile Configuration"（分析配置）中，选择 "C/C++ Oracle Solaris Studio Standard"（C/C++ Oracle Solaris Studio 标准）。单击 "Profile Configuration"（分析配置）列表旁边的 "..." 按钮。
6. 请注意，为 "C/C++ Oracle Solaris Studio Standard"（C/C++ Oracle Solaris Studio 标准）选择的工具包括 "Thread Usage"（线程使用情况）、"Memory Usage"（内存使用情况）和 "CPU Usage"（CPU 使用情况）。



注 - 如果选择 "C/C++ Simple (indicators only)"（C/C++ 简单（仅指示器））或 "C/C++ Simple Solaris (indicators only)"（C/C++ 简单 Solaris（仅指示器）），则仅显示 "Run Monitor"（运行监视器）窗口中的指示器。单击每个指示器窗口中的详细信息按钮将显示以下消息："Detailed information is not available in the current profile configuration"（详细信息在当前分析配置中不可用）。要查看详细信息，请在项目属性中选择其他 "Profile Configuration"（分析配置），然后再次运行该项目。

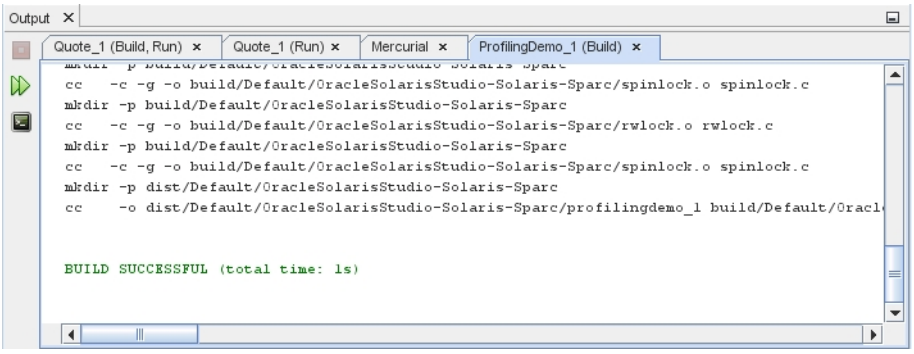
---

7. 在 "Project Properties"（项目属性）对话框中单击 "OK"（确定）。

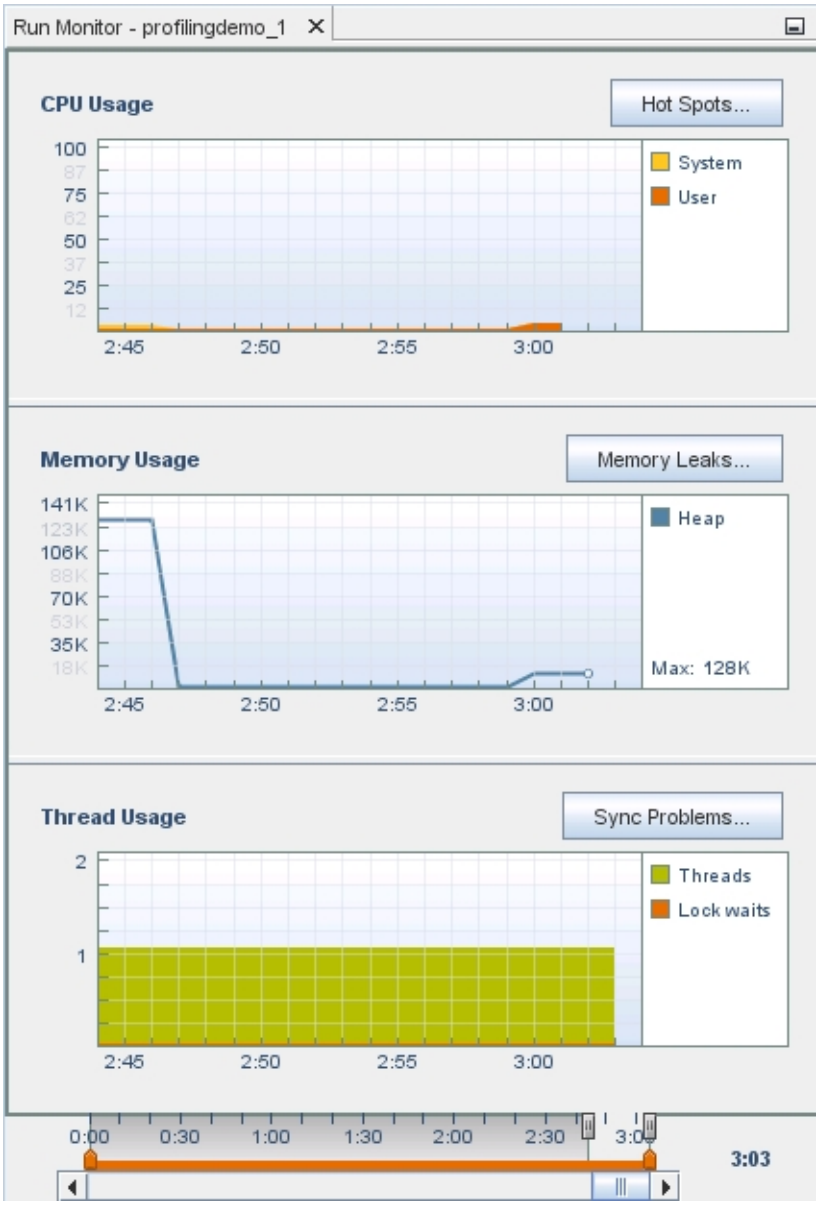
生成和运行 ProfilingDemo_1 项目

生成并运行 ProfilingDemo_1：

1. 右键单击 ProfilingDemo_1 项目节点，然后选择 "Build"（生成）。
   "Output"（输出）窗口显示生成的结果。



2. 右键单击 ProfilingDemo_1 项目节点，然后选择 "Run"（运行）。"Run Monitor"（运行监视器）窗口打开，显示带有动态图形的 "CPU Usage"（CPU 使用情况）、"Memory Usage"（内存使用情况）和 "Thread Usage"（线程使用情况）指示器。



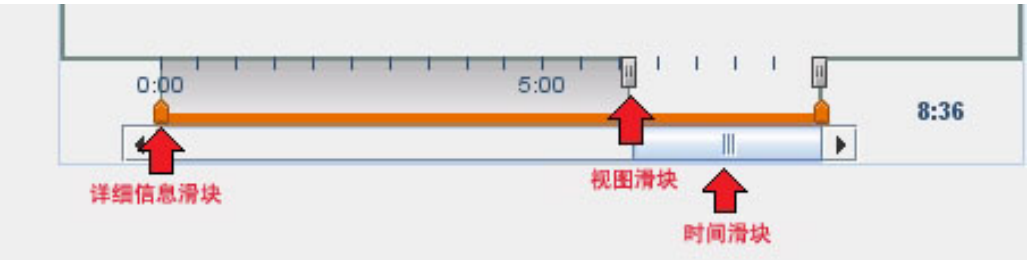
请注意，在输出窗口中，ProfilingDemo_1 程序将显示正在执行的操作，这样您就可以将其与 IDE 在工具中以图形表示的数据进行比较。例如，程序会显示其分配了多少内存，并且还会执行计算，然后释放内存。您可以看到该图反映了此活动。

3. 在每次出现提示时按 Enter 键，直至程序完成。

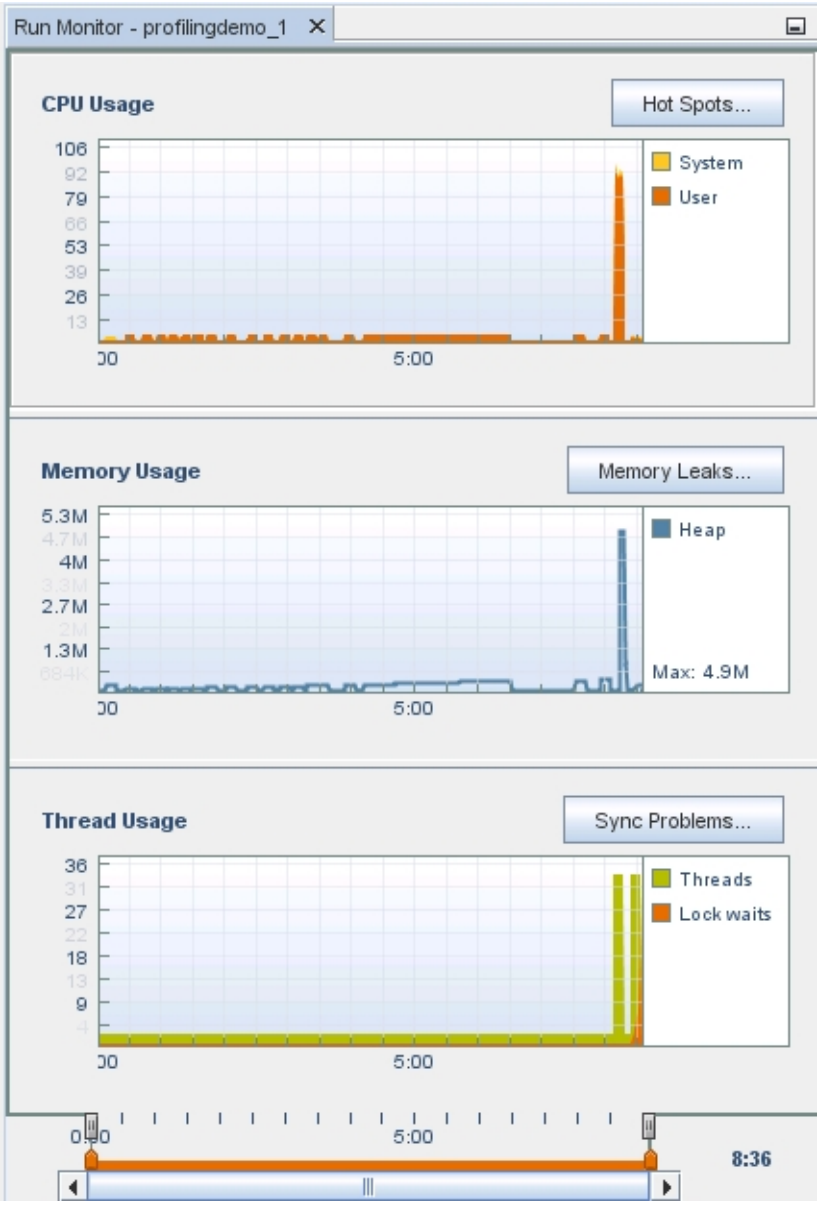
4. 将鼠标光标悬停在指示器上，以查看解释每个图形所表示内容的工具提示。每个指示器包含一个用于显示更详细信息的按钮，将在后面部分介绍该按钮。

使用指示器控件

在 "Run Monitor"（运行监视器）窗口底部，可以看到用于控制图形视图的滑块：视图滑块、详细信息滑块和时间滑块。这些控件还用于过滤数据。下图标记了这些控件。



1. 将鼠标光标放在滑块端点上以显示滑块的相关信息。
2. 在时间滑块（底部的水平滚动条）上单击并按住鼠标按钮。所有图形将一致地滑动，这样，您可以看到任何给定时间的 CPU、内存和线程，并可以查看它们之间的关系。



从左向右拖动时间滑块以查看完整运行情况。

3. 将鼠标移到视图滑块（与时间单位重叠的控件）。视图滑块控制在指示器中显示的运行时间部分。

4. 单击视图滑块起点的手柄并将其拖动到运行开始部分。指示器显示整个运行情况。效果与缩小操作类似。请注意，选择完整运行时间时会将时间滚动条最大化，因为您已经在查看所有数据。
5. 将视图滑块的起点拖到 PTHREAD_MUTEX_DEMO 开始的位置。在拖动手柄时，指示器将放大以聚焦到此区域。请注意，可以重新使用滚动条在运行时间中滚动。
6. 将鼠标光标放在橙色详细信息滑块终点上，以显示如何使用滑块的说明。通过使用详细信息滑块控件，可以选择某个运行时部分以查看详细信息。

将详细信息滑块的起点手柄拖过视图滑块的起点。请注意，指示器将详细信息滑块起点前面的区域灰显。这样，便可突出显示起点和终点之间的图形。单击指示器的详细信息按钮时，详细信息选项卡将显示突出显示的区域的数据。

7. 将详细信息滑块的起点拖回到开始处，这样可以看到所有数据。

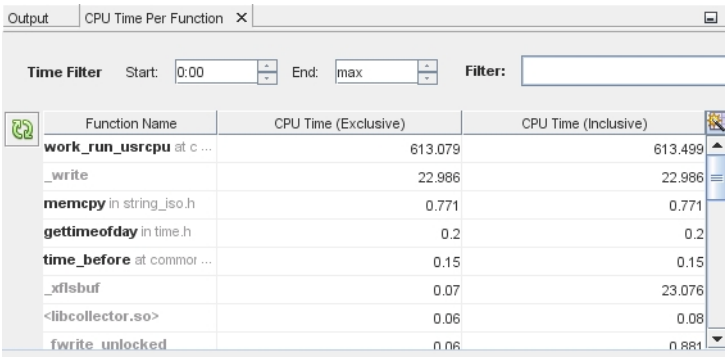
注 - 可以滑动视图滑块和详细信息滑块的起点手柄或终点手柄，从而在运行中随时查看和获取详细信息。

了解 CPU 使用情况

"CPU Usage"（CPU 使用情况）图显示了应用程序在其运行期间所使用的 CPU 总时间的百分比。

1. 单击 "CPU Usage"（CPU 使用情况）中的 "Hot Spots..."（热点...）按钮以显示有关 CPU 时间的详细信息。

   "CPU Time Per Function"（每个函数的 CPU 时间）窗口会打开，显示程序的各个函数以及每个函数所使用的 CPU 时间。这些函数按照使用的 CPU 时间顺序列出，使用时间最多的函数将列在最前面。如果程序仍在运行，最初显示的时间是单击图形时所用的时间。



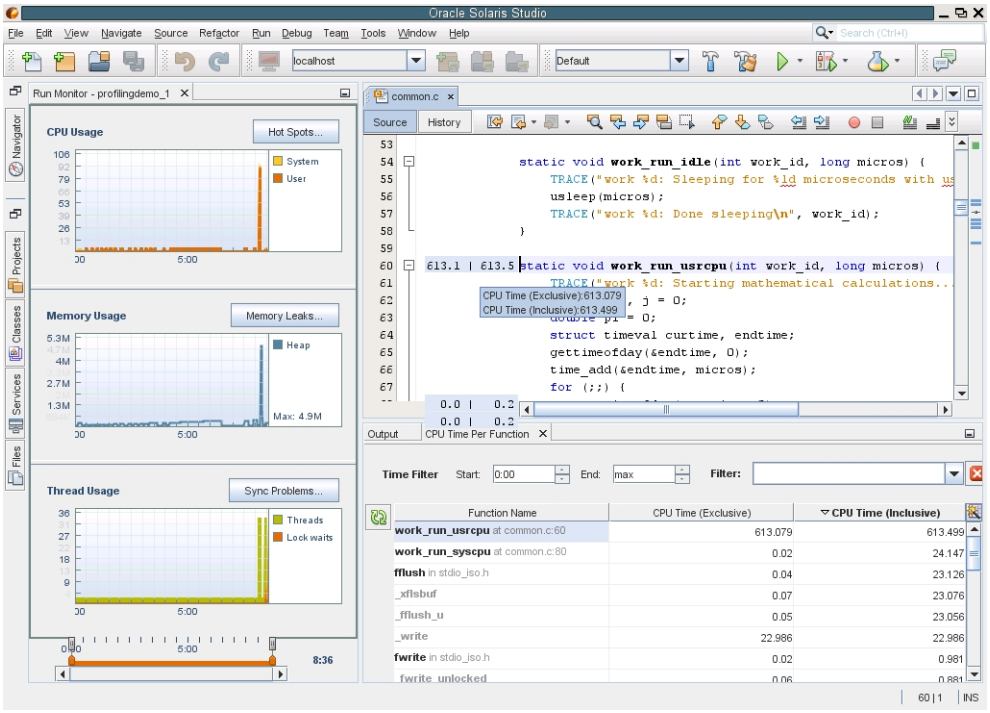
2. 单击 "Function Name"（函数名）列标题以按字母顺序对函数进行排序。

   请注意两列 "CPU Time"（CPU 时间）之间的差异。"CPU Time (Inclusive)"（CPU 时间（包含））显示从进入函数的时间起直到函数退出的时间为止所花费的 CPU 总时间，包括所列函数调用的所有其他函数的时间。"CPU Time (Exclusive)"（CPU 时间（独占））仅显示特定函数所使用的时间，而不包括它所调用的任何函数。

3. 单击 "CPU Time (Inclusive)"（CPU 时间（包含））列标题可将用时最多的函数放回到顶部。

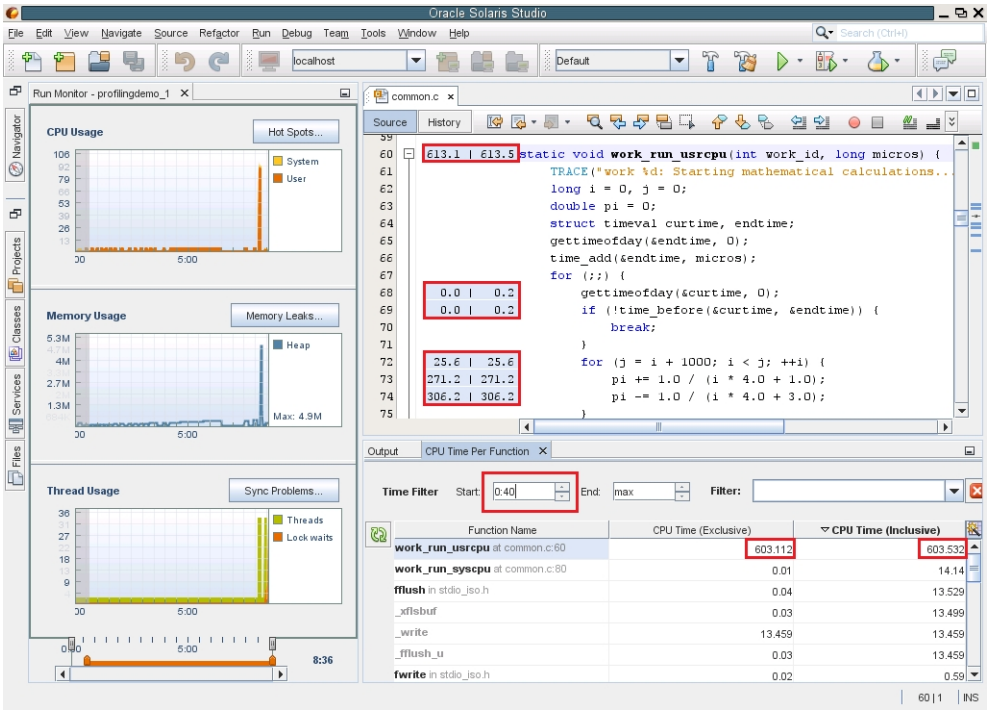
   请注意，在我们的示例中，`work_run_usrcpu（）` 函数的 "CPU Time (Exclusive)"（CPU 时间（专用））为 613.079，而 "CPU Time (Inclusive)"（CPU 时间（包含）） 为 613.499。这意味着 `work_run_usrcpu（）` 函数调用的其他函数实际使用了少量的 CPU 时间，而其自身使用了大部分时间。

4. 一些函数以粗体列出。可以转至调用这些函数的源文件。双击 `work_run_usrcpu（）` 函数。



将会打开 `common.c` 文件，并且光标停留在第 60 行的 `work_run_usrcpu（）` 函数上。此行的左边界中显示一些数字。

5. 将光标悬停左边界中的数字上。这些数字是与 "CPU Time Per Function"（每个函数的 CPU 时间）窗口中所显示函数的专用和包含 CPU 时间相同的度量。这些度量四舍五入到了小数点后一位，但将鼠标悬停在这些数字上时将弹出未四舍五入前的值。`common.c（）` 源文件中也会显示 CPU 消耗行的度量（例如在 `work_run_usrcpu（）` 函数内执行计算的 `for` 循环）。
6. 在 "CPU Time Per Function"（每个函数的 CPU 时间）窗口中，通过键入时间并按 Enter 键，或者使用箭头滚动选择秒数，将 "Time Filter"（时间过滤器）开始时间更改为 0:40。图形指示器将发生变化，如同在数据过滤控件上移动手柄时一样。如果拖动手柄，将更新 "CPU Time Per Function"（每个函数的 CPU 时间）窗口中的 "Time Filter"（时间过滤器）设置以保持一致。更重要的是，将更新为表中的函数显示的数据以反映过滤器设置，以便仅显示在该时间段内使用的 CPU 时间。



7. 也可以进行过滤以仅显示满足特定度量的数据。右键单击 `work_run_usrcpu（）` 的 "CPU Time (Exclusive)"（CPU 时间（独占））度量。在我们的示例中，右键单击 "CPU Time Per Function"（每个函数的 CPU 时间）窗口中的 603.112。选择 "Show only where > CPU Time (Exclusive) == 603.112"（仅显示 CPU 时间（专用）等于 603.112 的行）。此时将过滤所有行，并仅显示其专用 CPU 时间等于 603.112 的行。

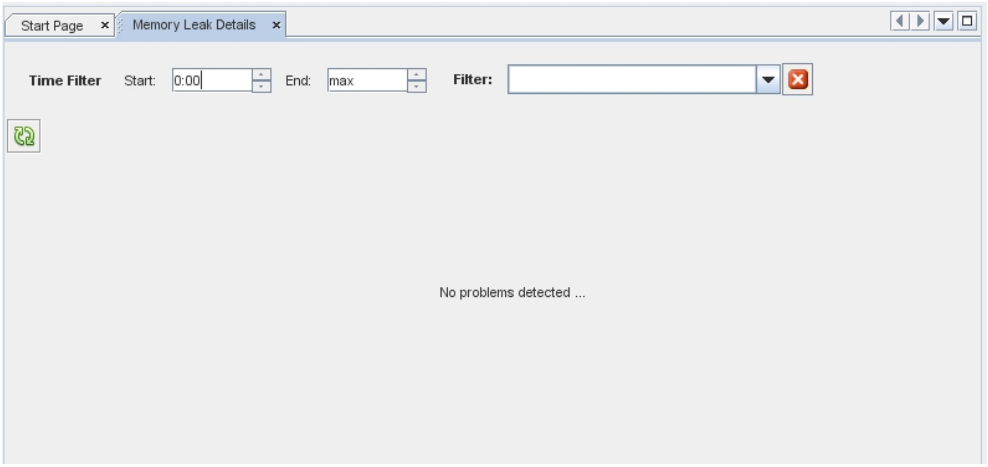
了解内存使用情况

"Memory Usage"（内存使用情况）指示器显示了项目的内存堆在其运行时如何变化。您可以使用该工具来识别内存泄漏，内存泄漏是指程序中不再需要的内存无法释放的点。内存泄漏可能会导致程序中的内存消耗增加。如果某个存在内存泄漏的程序运行足够长的时间，它最终可能会用尽可用内存。

1. 在 "Run Monitor"（运行监视器）中左右滑动滑块，以查看内存堆如何随时间增加或减少。ProfilingDemo_1 中应该出现四次使用峰值。前两次出现在串行演示期间，第三次出现在并行演示期间，最后一次出现在 Pthread 互斥演示期间。
2. 单击 "Memory Leaks"（内存泄漏）按钮显示 "Memory Leak Details"（内存泄漏详细信息）窗口。这将显示有关哪些函数呈现内存泄漏的详细信息。出现的表中只会

列出产生内存泄漏的函数。如果单击该按钮时程序正在运行，显示的泄漏位置就是单击按钮那一刻存在的位置。随着时间的推移，可能会发生更多泄漏，因此，应单击 "Refresh"（刷新）按钮。如果直到运行结束仍未检测到任何内存泄漏，"Memory Leak Details"（内存泄漏详细信息）选项卡会指示未找到内存泄漏。

3. 可以更改 "Start"（开始）和 "End"（结束）时间来过滤数据，或使用 "Run Monitor"（运行监视器）窗口中的橙色详细信息滑块（与“了解 CPU 使用情况”一节中相同）。本示例中没有内存泄漏。

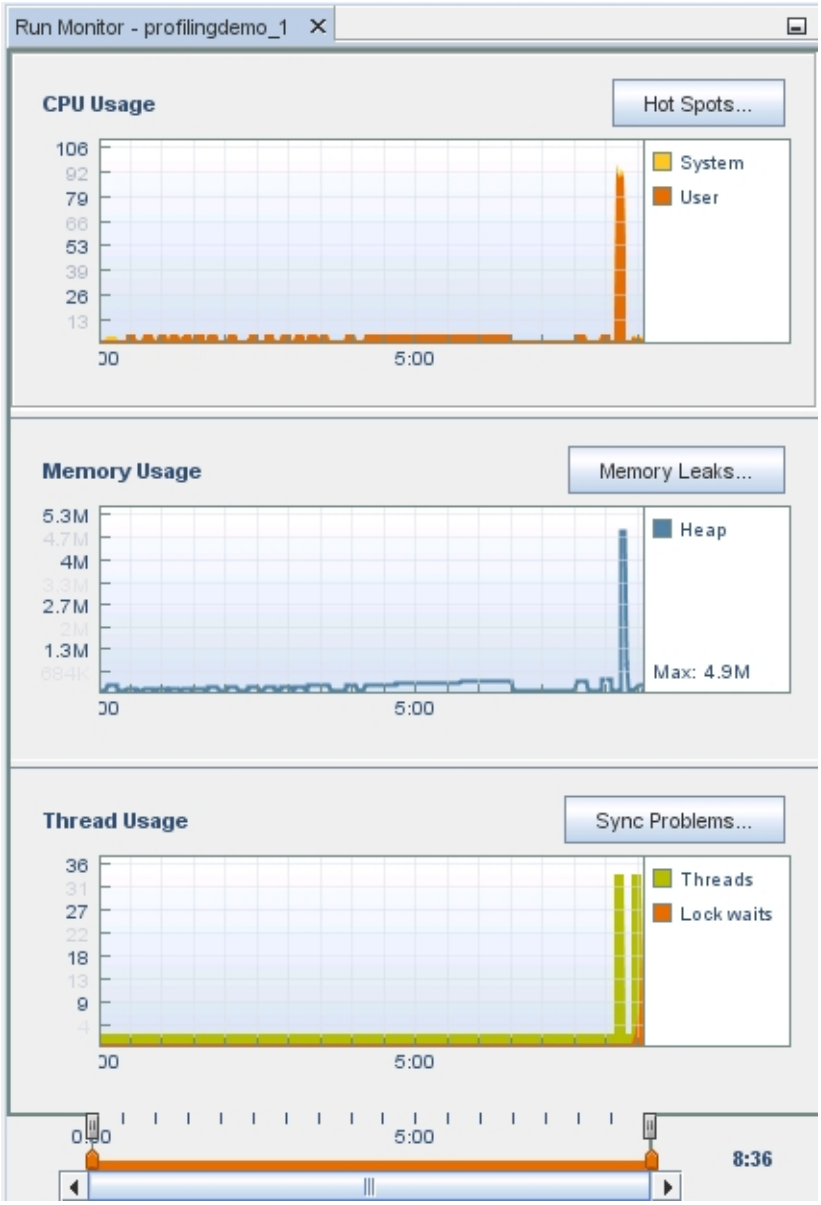


4. 双击某个函数，将打开相应文件并停在函数中发生内存泄漏的行。源代码编辑器的左边界中将显示内存泄漏度量。
5. 将鼠标移到度量上以显示详细信息，如同对 CPU 使用情况度量执行的操作一样。
6. 右键单击表中的度量以过滤表中的数据。此表中数据过滤功能在所有分析工具中均可用。

了解线程使用情况

"Thread Usage"（线程使用情况）指示器显示了程序正在使用的线程数量以及线程必须等待锁以继续执行其任务的时间。此数据对于多线程应用程序很有用，这些应用程序为了避免昂贵的等待时间必须执行线程同步。

1. "Thread Usage"（线程使用情况）指示器显示项目运行时所运行的线程数。将时间滑块滑回开始处，注意线程数为 1，直到并行演示开始。在下图中，该时间为 8:07，从 1 个线程跳跃到 32 个。



2. 将视图滑块端点手柄移至并行演示刚刚开始的位置。

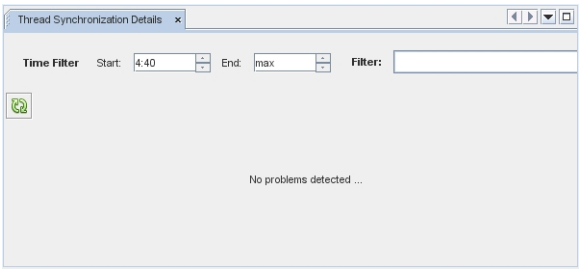
请注意，在同一时间段的 "CPU Usage"（CPU 使用情况）和 "Memory Usage"（内存使用情况）指示器中，单个线程正在执行某个使用 CPU 时间和内存的活动。此时间段对应于串行演示部分，主线程先在文件中写入数据，然后再执行一些计算。在程序等待用户按 Enter 键时，CPU 和内存使用率将会减少，线程数保持为 1 个。

3. 向右滑动时间滑块，直到看到线程数增加的两个点。

   8:07 时的线程数增加与项目运行的并行演示部分相对应。主线程启动其他线程，执行向文件写入的工作以及并行执行计算。请注意内存使用情况和 CPU 使用情况的增加，但是完成两个任务的时间要少得多。
4. 在并行演示线程结束后，线程数恢复为 1 个，主线程等待您按 Enter 键。运行程序的 Pthread 互斥演示部分时，线程数将再次增加。

   请注意，在 Pthread 互斥演示部分出现锁等待，以橙色显示。Pthread 互斥演示使用互斥锁来防止多个线程对某些函数进行重叠访问，这会导致线程需要等待获取锁。

   单击 "Sync Problems"（同步问题）按钮以显示有关项目中的线程锁定的详细信息。将打开 "Thread Synchronization Details"（线程同步详细信息）窗口，并列出必须等待获取互斥锁的函数。还会显示以下度量：函数等待时所花费的毫秒数以及函数必须等待获得锁的次数。本示例没有同步问题。



   如果在程序运行时单击 "Sync Problems"（同步问题）按钮，则可能需要单击刷新按钮  ，以使用最新线程锁定信息更新显示。
5. 单击 "Wait Time"（等待时间）列以按等待所花费的时间顺序对函数进行排序。
6. 单击 "Lock Waits"（锁等待次数）列以按函数中线程等待的次数对函数进行排序。
7. 双击某个函数以便在编辑器中打开其源文件并停在该函数负责锁定内存位置的位置。"Wait Time"（等待时间）和 "Lock Waits"（锁等待次数）度量显示在源文件的左侧列中。请将鼠标放在度量上以查看详细信息，它与 "Thread Synchronization Details"（线程同步详细信息）窗口中显示的内容是一致的。
8. 右键单击这些度量，并取消选定 "Show Profiler Metrics"（显示分析器度量）。这些度量将不会再显示在任何配置工具的程序编辑器中。要重新显示度量，请在 IDE 菜单栏中选择 "View"（视图），然后选择 "Show Profiler Metrics"（显示分析器度量）。

对项目运行内存访问检查

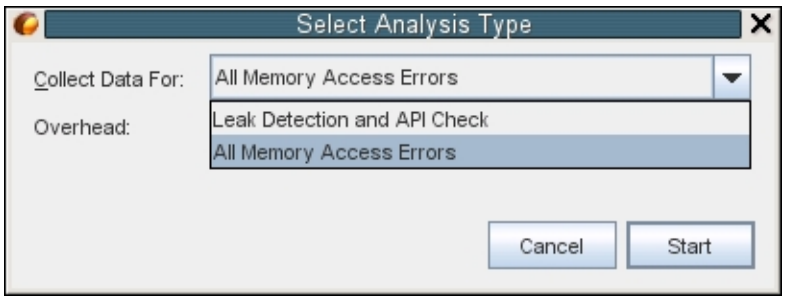
您可以使用内存分析工具查找项目中的内存访问错误。使用该工具，您可以定位源代码中出现各个错误的确切位置，从而轻松地找到这些错误。

内存分析工具会在程序执行期间动态捕捉并报告内存访问错误，因此，如果您的代码的某个部分在运行时未执行，则不会报告该部分的错误。

1. 如果您尚未执行此操作，请从位于 http://www.oracle.com/technetwork/server-storage/solarisstudio/downloads/solaris-studio-samples-1408618.html 的 Oracle Solaris Studio 12.3 Sample Applications（Oracle Solaris Studio 12.3 样例应用程序）Web 页下载样例应用程序 zip 文件，然后将该文件解压缩到您选择的位置中。`memorychecks` 应用程序位于 `SolarisStudioSampleApplications` 目录的 `CodeAnalyzer` 子目录中。
2. 使用 `memorychecks` 应用程序基于现有源代码创建项目。
3. 右键单击项目，然后选择 "Properties"（属性）。在 "Project Properties"（项目属性）对话框中，选择 "Run"（运行）节点，然后在 "Run Command"（运行命令）的输出路径后面键入 Customer.db。单击 "OK"（确定）。
4. 运行项目。
5. 在采用内存分析检测的情况下生成项目。

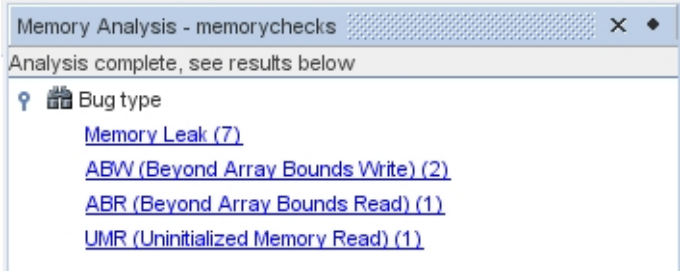
   a. 单击 "Profile Project"（分析项目）按钮  旁边的向下箭头，然后从下拉式列表中选择 "Profile Project to find Memory Access Errors"（分析项目以查找内存访问错误）。

   b. 在 "Select Analysis Type"（选择分析类型）对话框中，从下拉式列表中选择 "All Memory Access Errors"（所有内存访问错误）。

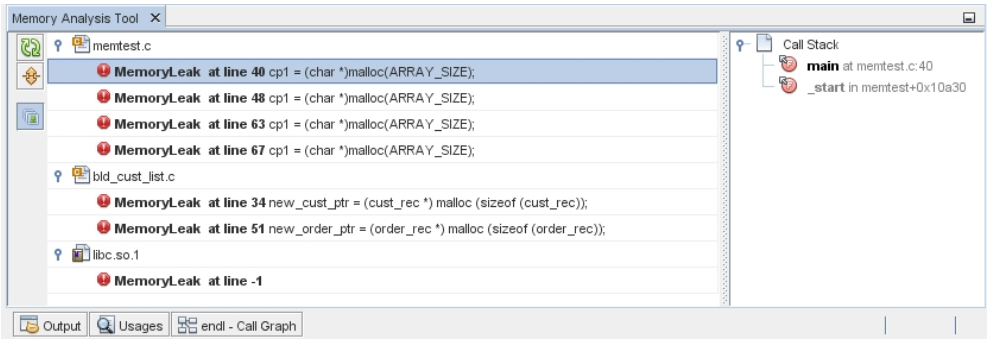


   "Overhead"（开销）字段中会显示 "High"（高）或 "Moderate"（中等）来表示系统将承受的负载。当系统开销很大时（例如，您同时检测数据争用和死锁时），在系统上运行的其他程序的性能可能会受到影响。

   c. 单击 "Start"（开始）。
6. 会打开 "Run Memory Profile"（运行内存分析）对话框，让您知道将对您的二进制文件进行检测。单击 "OK"（确定）。
7. 此时将生成并检测项目。应用程序开始运行并且 "Memory Analysis"（内存分析）窗口将打开。项目运行完成时，"Memory Analysis"（内存分析）窗口会列出在项目中找到的内存访问错误类型。在错误类型后面会在括号中显示每种类型的错误数。



8. 单击某一错误类型时，该类型的错误将显示在 "Memory Analysis Tool"（内存分析工具）窗口中。



缺省情况下，错误按它们所在的源文件分组。单击某一错误时，将显示该错误的调用堆栈。双击堆栈中的某个函数调用可显示源文件中的相关行。

♦ ♦ ♦ 第 9 章

9 执行远程开发

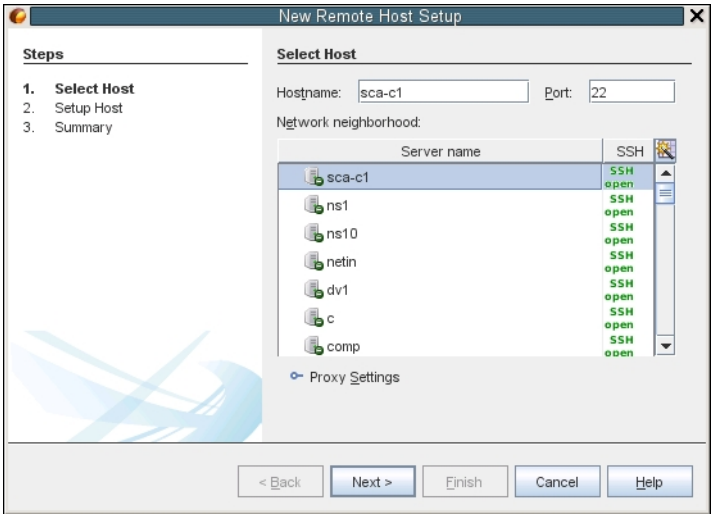
本章介绍如何执行远程开发。同时还介绍了如何使用远程开发工具栏。另外，本章简要描述了远程桌面分发。

执行远程开发

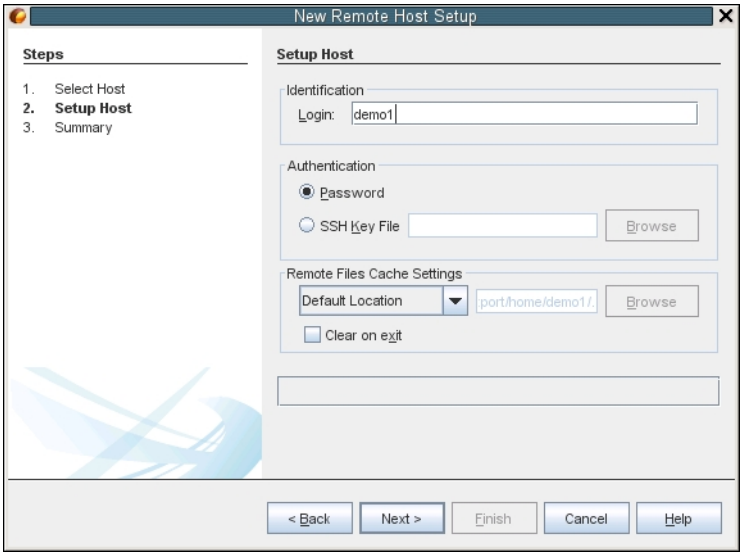
您可以在本地主机（从中启动 IDE 的系统）上生成、运行和调试项目，也可以在运行 UNIX® 操作系统的远程主机上生成、运行和调试项目。借助 "Remote development"（远程开发），您可以在熟悉的桌面环境中本地运行 IDE，同时使用远程服务器的计算能力和开发工具来生成您的项目。

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

1. 选择 "Tools"（工具）> "Options"（选项），然后单击 "C/C++" 类别。
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 "Build Hosts Manager"（生成主机管理器）对话框中，单击 "Add"（添加）。
4. 在 "New Remote Build Host"（新建远程生成主机）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. 在主机添加到 "Build Hosts Manager"（生成主机管理器）对话框中的 "Build Hosts"（生成主机）列表中后，单击 "OK"（确定）。

8. 您可以在 "Services"（服务）窗口中设置用来指定 IDE 如何使用远程主机的属性。展开 "C/C++ Build Hosts"（C/C++ 生成主机）节点，右键单击远程主机，然后选择 "Properties"（属性）。在 "Host Properties"（主机属性）对话框中设置所需的属性。
9. 要将远程主机设置为缺省生成主机，请在 "Services"（服务）窗口中右键单击 "C/C++ Build Hosts"（C/C++ 生成主机）节点中的主机，然后选择 "Set as Default"（设置为缺省）。

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。

在创建项目时，缺省生成主机将被选作项目的生成主机。可以在 "Project Properties"（项目属性）对话框的 "Build"（生成）面板上更改项目的生成主机。还可以在调试可执行文件或信息转储文件时指定生成主机。

要在本地主机上处理位于远程主机上的项目，请选择 "File"（文件）> "Open Remote C/C++ Project"（打开远程 C/C++ 项目）。

使用远程开发工具栏

IDE 提供了远程开发工具栏，以便更轻松地访问远程主机上的项目和文件。您可以使用图标来选择已配置的远程主机并处理远程主机上的项目和文件，就如在本地一样。这称为完全远程开发。

要了解远程开发模式的更多信息，请参见 IDE 中的 "C/C++ Remote Development Modes"（C/C++ 远程开发模式）帮助页。

可以使用远程工具栏执行以下操作：

- 单击 "Connect Status"（连接状态）图标，连接到该图标旁列表中选定的服务器或断开与该服务器的连接。
- 单击 "Create Remote Project"（创建远程项目）图标，在当前连接的主机上创建一个新项目。
- 单击 "Open Remote Project"（打开远程项目）图标，在当前连接的主机上打开项目。
- 单击 "Open Remote File"（打开远程文件）图标，在当前连接的主机上打开文件。

使用生成主机的终端窗口

您可以在 IDE 中为本地主机或远程主机打开终端窗口。终端窗口作为 IDE 的 "Output"（输出）区域中的选项卡打开。无需提前为 IDE 设置远程主机，但是必须运行 SSH 守护进程。

要了解如何在 IDE 中打开终端窗口的信息，请参见 IDE 帮助中的 "Opening a Terminal Window for a Build Host"（为生成主机打开终端窗口）。要为本地主机打开终端窗口，请选择 "Window"（窗口）> "IDE Tools"（IDE 工具）> "Terminal"（终端）。

还可以个性化命令行终端窗口的显示选项。有关更多信息，请参见 IDE 帮助中的 "Options Window: Miscellaneous: Terminal"（选项窗口：其他：终端）。

远程桌面分发

通过“远程桌面分发”功能可以生成一个 zip 文件，其中包含 IDE 和代码分析器的分发，该分发几乎可在所有操作系统上运行，该功能还可在远程服务器上使用 Oracle Solaris Studio 编译器和工具。在桌面系统上运行 IDE 时，它将生成分发的服务器视为远程主机，并访问 Oracle Solaris Studio 安装中的工具集合。

有关远程桌面分发的更多信息，请参见 How to Develop Code from a Remote Desktop with Oracle Solaris Studio (http://www.oracle.com/technetwork/articles/servers-storage-dev/howto-use-ide-desktop-1639741.html)（如何使用 Oracle Solaris Studio 基于远程桌面开发代码）。

♦ ♦ ♦ 第 10 章

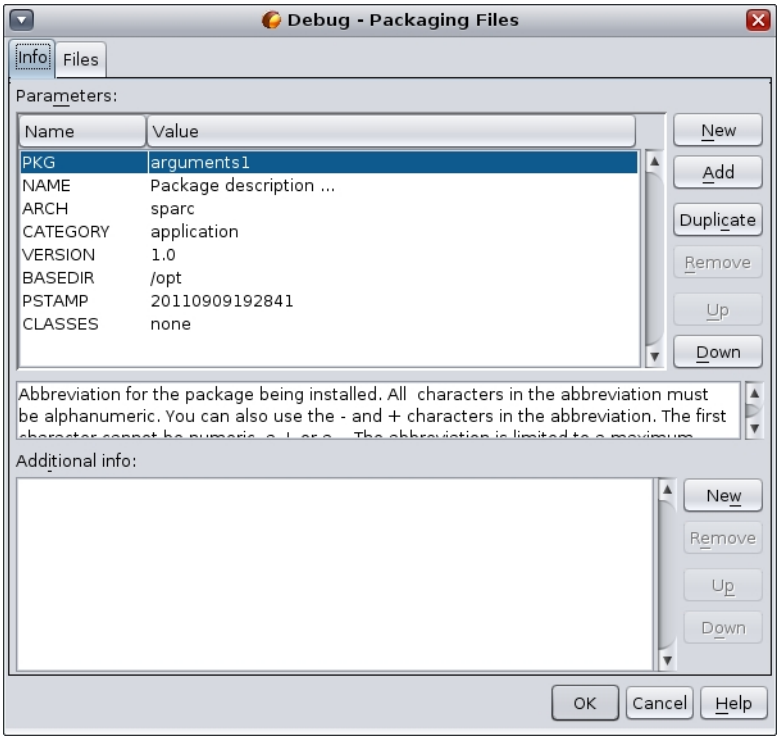
10 将应用程序打包

本章介绍了在完成应用程序的创建、编译和调试后如何将应用程序打包。

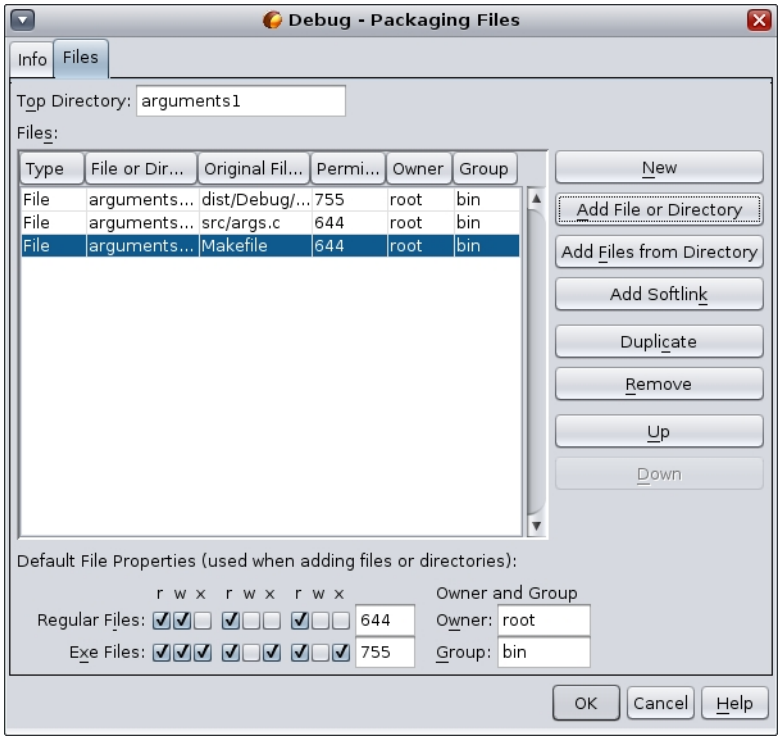
将应用程序打包

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）。

♦ ♦ ♦ 第 11 章

11 有关 IDE 的更多信息

本章将带您了解有关 Oracle Solaris Studio IDE 的其他资源。

了解有关 IDE 的更多信息

您可以阅读 IDE 内集成的帮助，了解有关使用 Oracle Solaris Studio IDE 的信息。可以通过 IDE 的 "Help"（帮助）菜单或按 F1 键来访问该帮助。很多对话框中也都带有 "Help"（帮助）按钮，可以用来了解有关如何使用相应对话框的信息。

此外，尽管在用户界面和功能之间有一些差异，NetBeans IDE C/C++ 学习资源上的教程也可以帮助了解如何使用 Oracle Solaris Studio IDE。需要特别说明的是，关于调试的 NetBeans 文档不适用于 Oracle Solaris Studio IDE。

Oracle 数据库项目

有关 IDE 中的数据库资源管理器的更多信息，请参见 How to Use the Database Explorer in the Oracle Solaris Studio IDE（如何在 Oracle Solaris Studio IDE 中使用数据库资源管理器）。

有关 IDE 中的 Oracle Instant Client 的信息，请参见 How to Use Oracle Instant Client in the Oracle Solaris Studio IDE（如何在 Oracle Solaris Studio IDE 中使用 Oracle Instant Client）。

远程开发

有关选择远程开发模式的信息，请参见 How to Select a Remote Development Mode in Oracle Solaris Studio（如何在 Oracle Solaris Studio 中选择远程开发模式）。

有关远程开发的更多常规信息，请参见 IDE 中的 "Overview of C/C++ Remote Development"（C/C++ 远程开发概述）帮助页。此页还将带您进入有关远程开发的其他有用页面。

版本控制系统

IDE 集成了用于使用多个版本控制系统（如 Mercurial、Subversion、CVS 和 Git）的选项，从而使项目的源代码管理更加轻松。有关更多信息，请参见 How to Use IDE Projects Under Version Control（如何使用版本控制下的 IDE 项目）。