Oracle® Developer Studio 12.6：概述

文件号码 **E83256-01**
2017 年 7 月

ORACLE

文件号码 E83256-01

文档可访问性

有关 Oracle 对可访问性的承诺，请访问 Oracle Accessibility Program 网站 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc。

获得 Oracle 支持

购买了支持服务的 Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

目录

使用本文档 7

Oracle Developer Studio 12.6 概述 9
Oracle Developer Studio 软件介绍 9
Oracle Developer Studio 的开发者工作流程 10
Oracle Developer Studio IDE 11
Oracle Developer Studio 编译器 14
C 编译器 14
C++ 编译器 15
Fortran 95 编译器 17
C/C++/Fortran 库 18
用于并行编程的 OpenMP 4.0 18
适合密集计算程序的 Oracle Developer Studio 性能库 18
Oracle Developer Studio 代码安全性检查－Discover ADI 19
用于生成应用程序的 `dmake` 实用程序 20
用于调试应用程序的工具 20
命令行中的 `dbx` 21
IDE 中的 `dbx` 22
`dbxtool` 中的 `dbx` 23
用于验证应用程序的工具 24
用于检测内存错误的 `discover` 工具 25
用于度量代码覆盖的 `uncover` 工具 26
用于集成错误检查的代码分析器工具 27
用于集成检查的 `codean` 工具 28
用于调整应用程序性能的工具 28
性能分析器工具 28
IDE 中的分析工具 34
有关更多信息 37

使用本文档

- **概述**－介绍了 Oracle Developer Studio 产品中提供的工具、编译器和编程库，并说明了如何在开发工作流程中将它们组合使用。
- **目标读者**－应用程序开发者、系统开发者、架构师、支持工程师
- **必备知识**－无

产品文档库

有关该产品及相关产品的文档和资源，可从以下网址获得：http://www.oracle.com/pls/topic/lookup?ctx=E83242-01。

反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关本文档的反馈。

Oracle Developer Studio 12.6 概述

Oracle Developer Studio 软件是针对 Oracle Solaris™ 和 Linux 平台上 C、C++ 和 Fortran 开发的一套软件开发工具，支持使用 SPARC® 处理器和 x86 以及 x64 处理器的多核系统。

Oracle Developer Studio 软件介绍

Oracle Developer Studio 由两套工具组成：编译器套件和分析套件。每个套件中所包含的工具都设计为相互配合使用，为单线程、多线程和分布式应用程序的开发提供优化的开发环境。

开发在 SPARC 或 x86 和 x64 平台上的 Oracle Solaris 10 或 Oracle Solaris 11 中运行的 C、C++ 和 Fortran 应用程序时，或者开发在 x86 和 x64 平台上的 Oracle Linux 中运行的 C、C++ 和 Fortran 应用程序时，Oracle Developer Studio 可提供所需的任何内容。编译器和分析工具的设计使您的应用程序能在 Oracle Sun 系统中以最理想的状态运行。

具体而言，Oracle Developer Studio 编译器和分析工具的设计保证了它们可以利用较新的多核 CPU（包括 SPARC T5、SPARC M5、SPARC M6、SPARC M7、SPARC M10、SPARC M10+ 以及 Intel Ivy Bridge 和 Haswell 处理器）的功能，也可利用较早的 SPARC T4、SPARC T3 以及 Intel® Xeon® 和 AMD Opteron™ 处理器的功能。利用 Oracle Developer Studio 可以更轻松地创建适用于这些平台的并行和并发软件应用程序。

Oracle Developer Studio 的组件包括：

- IDE，用于在图形环境中开发应用程序。Oracle Developer Studio IDE 集成了多种其他 Oracle Developer Studio 工具。
- C、C++ 和 Fortran 编译器，用于使用命令行或通过 IDE 来编译代码。这些编译器设计为与 Oracle Developer Studio 调试器 (`dbx`) 配合使用，并包含一些可以用来针对特定处理器优化代码的选项。
- 各种库，可以提高应用程序的高级性能和多线程处理能力。
- Make 实用程序 (`dmake`)，用于在分布式计算环境中使用命令行或通过 IDE 构建代码。
- 调试器 (`dbx`)，用于使用命令行或通过 IDE 或者通过独立的图形界面 (`dbxtool`) 来发现代码中的错误。

- 代码分析器工具，用于发现编译期间代码中的静态代码错误，以及执行期间的内存访问和代码覆盖错误。
- 性能分析器工具，采用 Oracle Solaris 技术（如 DTrace），可以在命令行或通过图形界面使用，用来发现无法通过调试检测到的代码中的故障点。
- 线程分析器，用于检查多线程程序，以检测可导致数据争用和死锁的编程错误。

您可以将这些工具结合使用，以生成、调试及调整您的应用程序，从而在运行于 Oracle Sun 系统上的 Oracle Solaris 中实现高性能。本文档的后面部分对每个组件进行了更详尽的介绍。

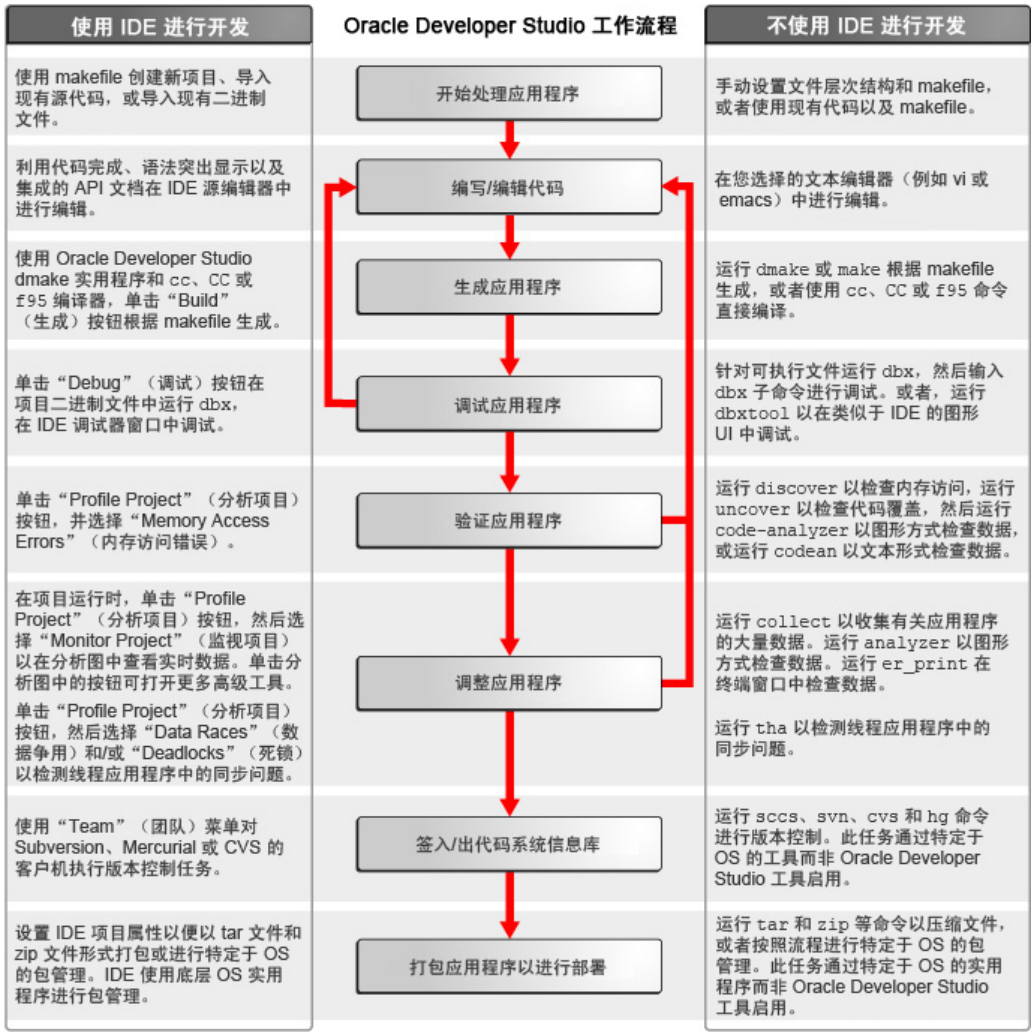
Oracle Developer Studio 的开发者工作流程

Oracle Developer Studio 提供了各种工具来帮助开发者创建在 Oracle Solaris 上运行的应用程序。这些工具可以为需要使用图形 IDE 管理大量开发任务的开发者以及希望使用自己的方法对软件开发的所有方面进行控制的开发者提供支持。

您并非必须使用 IDE 或命令行，因为这些工具设计为可以任意组合使用。可以在 IDE 中创建项目，而从命令行使用 `dmake` 或 `make` 来编写项目的源代码。您可以对 IDE 中所创建项目的二进制文件使用性能分析器。IDE 使其项目文件与源文件相分离，因而不具有相关性。

如果您是忠实的 Emacs 或 vi 用户，可以继续使用惯用的环境而忽略 IDE，但可以采用 Oracle Developer Studio 编译器和性能工具，以使应用程序在使用 Oracle Sun 硬件环境的 Oracle Solaris 中以最理想状态运行。

下图说明了在使用或不使用图形 IDE 的情况下，Oracle Developer Studio 工具的开发者工作流程。



本文档的其余内容介绍 Oracle Developer Studio 软件的组件，阐述这些组件的集成方式，并简要说明如何使用这些组件。

Oracle Developer Studio IDE

Oracle Developer Studio IDE 以开源的集成式开发环境 NetBeans IDE 为基础。Oracle Developer Studio IDE 包含核心 NetBeans IDE、NetBeans C/C++ 插件模块以及开源 NetBeans IDE 中未提供的其他集成 Oracle Developer Studio 组件。

Oracle Developer Studio IDE 使用 Oracle Developer Studio C、C++ 和 Fortran 编译器、dmake 生成实用程序以及 dbx 调试器。此外，IDE 还提供图形分析工具，这些工具以不可见的方式使用 Oracle Developer Studio 性能实用程序来收集运行中项目的数据。

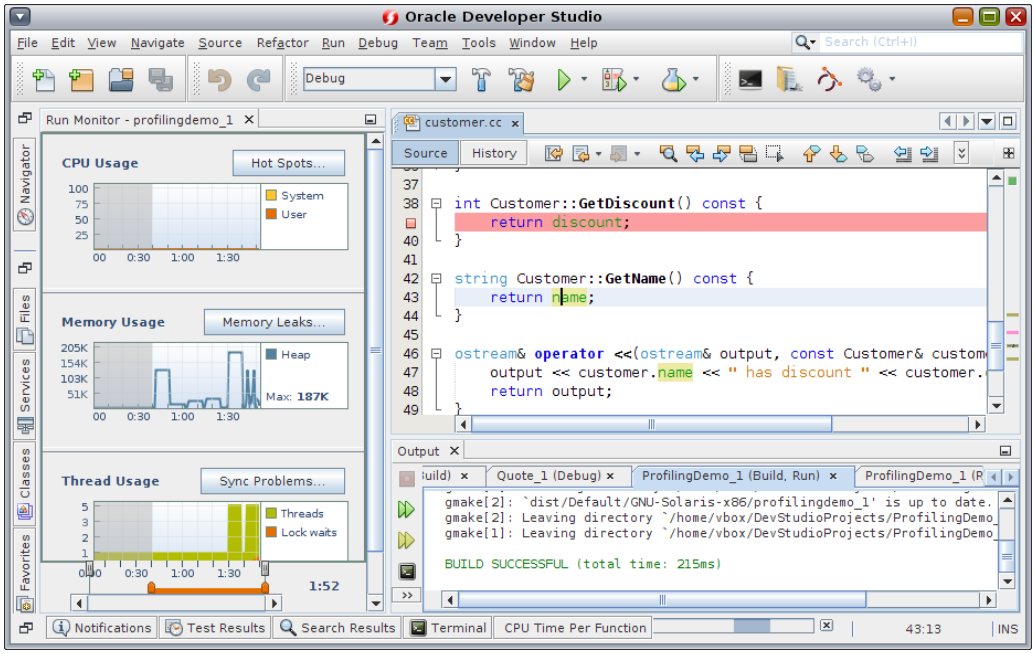
与使用文本编辑器和命令行进行开发相比，使用 Oracle Developer Studio IDE 具有以下优点：

- **代码编辑**。利用语法突出显示、代码完成、代码元素间导航以及集成的 API 文档和手册页，可以更高效地处理代码。
- **代码研究**。当您努力熟悉一些代码或查找错误的根源时，您可能会发现一些非常有用的 IDE 功能，例如 "Go to Symbol"（转至符号）、"Find Usages"（查找使用实例）、"Classes"（类）窗口、"Include Hierarchy"（包含分层结构）以及 "Call Graph"（调用图形）等。
- **重构**。您可以查找某个变量或操作在项目内的所有使用实例、对出现的所有变量或操作重新命名并在整个产品内进行重构。在执行重构之前，可以在分屏 UI 中预览更改，并对更改逐个进行批准或一次全部批准。
- **远程开发**。在使用安装在远程服务器上的 Oracle Developer Studio 编译器和工具的过程中，可以在桌面系统上运行 IDE、dbxtool、代码分析器和性能分析器。这些工具在远程服务器上运行，同时显示给在本地系统上运行的 IDE 或其他工具，与传统的远程显示解决方案相比，响应时间更短。

要启动 IDE，请键入以下命令：

```
% devstudio
```

下图说明了使用 Monitor Project 分析工具运行 Quote 样例项目的 IDE。



C、C++ 或 Fortran 项目是一组源文件，其中包含有关如何编译及链接这些源文件并运行生成的程序的信息。在 IDE 中，即使您的程序包含在单独的源文件中，您也始终是在项目内部工作。IDE 在包含 makefile 和元数据文件的项目文件夹中存储项目信息。源目录无需实际位于项目文件夹中。

每个项目（从现有二进制文件创建的项目除外）都必须有一个 makefile，以便 IDE 能够构建项目。项目的 makefile 可以由 IDE 生成，您也可以使用以前在 IDE 外部创建的 makefile。您可以从已经包含 makefile 的现有源文件，或者在您运行配置脚本时构建 makefile 的现有源文件创建项目。

您可以通过单击工具栏按钮或选择菜单命令来生成、运行及调试项目。缺省情况下，IDE 预配置为使用 Oracle Developer Studio C、C++ 和 Fortran 编译器 `dmake` 和 `dbx`。但是，如果系统上有 GNU 编译器，这些编译器在您的 PATH 中时，IDE 通常可以找到它们。您可以在项目属性中设置 "Tool Collection"（工具集合）来使用 GNU 工具集合。

您可以阅读 IDE 内集成的帮助，了解有关使用 Oracle Developer 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。

《Oracle Developer Studio 12.6：IDE 快速入门教程》介绍如何开始使用 IDE。此外，尽管在用户界面和功能之间有一些差异，关于 NetBeans IDE 的教程 C/C++ Learning Trail（C/C++ 学习资源）也可以帮助了解如何使用 Oracle Developer Studio IDE。需要特别说明的是，关于调试的 NetBeans 文档不适用于 Oracle Developer Studio IDE。

Oracle Developer Studio 编译器

Oracle Developer Studio 软件包含 C、C++ 和 Fortran 编译器，这些编译器具有以下功能：

- 符合现代 C、C++ 和 Fortran 编程语言标准。
- 根据指定的命令行选项，生成面向特定操作系统、处理器、体系结构、内存模型（32 位和 64 位）、浮点算法等等的代码。
- 对串行源代码执行自动并行化处理，以生成在多核系统中性能得到增强的二进制文件。
- 生成以您通过命令行选项指定的方式进行优化的代码，以适合应用程序和部署环境。
- 准备二进制文件，用于通过其他 Oracle Developer Studio 工具进行增强的调试或分析。
- 可以在所有编译器中使用相同的命令行选项来指定这些功能。

某些 Oracle Developer Studio 编译器选项可以用来优化编译后的代码以提高速度，并最佳利用处理器指令集和功能，这些选项包括：

`-O`*n*	指定由 *n* 所指示的优化级别，可以是从 1 到 5 的数字。优化级别越高，所创建的二进制文件的运行时性能越好。
`-fast`	针对可执行代码的速度选择最优的编译选项组合。在调整可执行代码以实现最佳运行时性能时，可以从有效地使用 `-fast` 选项开始。
`-g`	在二进制文件中生成更多针对使用 `dbx` 进行调试以及使用性能分析器进行分析的信息。使用 `-g` 选项进行编译时，您可以充分利用性能分析器的全部功能，如查看带注释的源代码、函数信息以及描述编译器在编译程序时执行的优化和转换的编译器注释消息。

与其他编译器相比，Oracle Developer Studio 编译器提供的信息明显增多，更有助于您理解代码。通过优化，这些编译器插入了注释，用来描述对代码执行的转换、对并行处理的阻碍、循环迭代的操作计数等等。编译器注释可以显示在性能分析器等工具中。

C 编译器

Oracle Developer Studio C 编译器符合 *ISO/IEC 9899:2011*、*ISO/IEC 9899:1999* 以及 *ISO/IEC 9899:1990* 编程语言－*C* 标准。C 编译器还支持 OpenMP 4.0 共享内存并行 API。

C 编译系统由一个编译器、一个汇编程序和一个链接程序组成。`cc` 命令会自动调用其中每个组件，除非您使用命令行选项单独执行这些步骤。

cc 命令语法

cc 命令的语法如下：

```
cc [compiler-options] source-files [-Ldir] [-llibrary]...
```

可以键入 `cc -flags` 以查看所有可能的编译器选项的简短说明。

源文件名称可以用 .c、.s、.S 或 .i 来结尾。名称不以其中的一个后缀结束的文件将传递到链接编辑器中。

还可以指定 -L*dir* 选项以将目录添加到列表中（以便链接程序搜索库），并可指定 -l*library* 选项以将对象库添加到链接程序的搜索库列表中。由 -L 选项指定的目录按所列的顺序进行搜索。

缺省情况下，链接编辑器生成名为 a.out 的动态链接可执行文件。可以使用 -o *filename* 选项指定其他可执行文件名称。可以使用 -c 选项编译源文件并生成目标 (.o) 文件，但隐藏链接。

编译名为 test.c 的源文件并生成名为 a.out 的可执行文件：

```
% cc test.c
```

编译源文件 test1.c 和 test2.c，并将这两个文件链接到称为 test 的可执行文件中：

```
% cc -o test test1.c test2.c
```

分别编译这两个源文件并将它们链接到一个可执行文件中：

```
% cc -c test1.c
% cc -c test2.c
% cc test1.o test2.o
```

C 文档

有关使用 C 编译器以及 cc 命令及其选项的完整信息，请参见《Oracle Developer Studio 12.6：C 用户指南》和 cc(1) 手册页。有关新功能和已更改功能的信息，请参见《Oracle Developer Studio 12.6 发行版的新增功能》。有关编译器的问题和解决方法以及限制和不兼容性的信息，请参见《Oracle Developer Studio 12.6：发行说明》。

C++ 编译器

Oracle Developer Studio C++ 编译器 (CC) 支持 *C++ ISO* 国际标准 *ISO/IEC 14882:2014* 编程语言－*C++*、*C++ ISO* 国际标准 *ISO IS 14822:2003* 编程语言－*C++* 和 *C++ ISO* 国际标准 *ISO/IEC 14882:2011*。CC 编译器还支持 OpenMP 4.0 共享内存并行 API。OpenMP 4.0 API 随 Oracle Developer Studio 12.6 提供。

C++ 编译器 (CC) 根据指定的命令行选项，生成面向特定操作系统、处理器、体系结构、内存模型（32 位和 64 位）、浮点算法等等的代码。该编译器会自动将序列源代码并行化以生成在多核系统上有更好性能的二进制文件，并且还可以准备二进制文件以便其他 Oracle Developer Studio 工具更好地进行调试或分析。该编译器还支持 GNU C/C++ 兼容功能。

C++ 编译器由前端、优化器、代码生成器、汇编程序、模板预链接程序和链接编辑器组成。CC 命令会自动调用其中每个组件，除非使用命令行选项进行其他指定。

CC 命令语法

CC 命令的语法如下：

CC [*compiler-options*] *source-files* [-L*dir*] [-l *library*]...

可以键入 CC -flags 以查看所有可能的 CC 编译器选项的简短说明。

源文件名称可以用 .c、.C、.cc、.cxx、.c++、.cpp 或 .i 结尾。名称不以其中的一个后缀结束的文件将视为目标文件或库，并移交给链接编辑器。

在源文件名称后，可以根据需要指定 -L*dir* 选项以将目录添加到链接程序的搜索库列表中，并可指定 -l*library* 选项以将对象库添加到链接程序的搜索库列表中。在命令行上，-L 选项必须在关联的库之前。

缺省情况下，将按照指定的顺序来编译和链接文件，以生成名为 a.out 的输出文件。可以使用 -o *filename* 选项指定其他可执行文件名称。可以使用 -c 选项编译源文件并生成目标 (.o) 文件，但隐藏链接。

编译名为 test.C 的源文件并生成名为 a.out 的可执行文件：

```
% CC test.C
```

分别编译两个源文件 test1.c 和 test2.C，然后将它们链接到称为 test 的可执行文件中：

```
% CC -c test1.c
% CC -c test2.C
% CC -o test test1.o test2.o
```

C++ 文档

有关使用 C++ 编译器以及 CC 命令及其选项的完整信息，请参见《Oracle Developer Studio 12.6：C++ 用户指南》和 CC(1) 手册页。有关新功能和已更改功能的信息，请参见《Oracle Developer Studio 12.6 发行版的新增功能》。有关编译器的问题和解决方法以及限制和不兼容性的信息，请参见《Oracle Developer Studio 12.6：发行说明》。

Fortran 95 编译器

Oracle Developer Studio 中的 Fortran 编译器已针对多处理器系统上的 Oracle Solaris 进行了优化。该编译器既可以执行自动循环并行，也可以执行显式循环并行，从而使程序能够在多处理器系统上高效运行。

Fortran 编译器与 Fortran77、Fortran90 和 Fortran95 标准兼容，并支持 OpenMP 4.0。

`f95` 命令调用 Oracle Developer Studio Fortran 编译器。

`f95` 命令语法

`f95` 命令的语法如下：

`f95 [`*compiler-options*`] `*source-files*`... [-l`*library*`]`

编译器选项附加在源文件名称前面。可以键入 `f95 -flags` 以查看所有可能的编译器选项的简短说明。

源文件名称必须是以 `.f`、`.F`、`.f90`、`.f95`、`.F90`、`.F95` 或 `.for` 结尾的一个或多个 Fortran 源文件的名称。

在源文件名称后，可以指定 `-l` *library* 选项（可选），以在链接程序的搜索库列表中添加对象库。

以下是通过两个源文件编译 Fortran 程序的样例命令：

```
% f95 -o hello_1 foo.f bar.f
```

使用单独的编译和链接步骤编译相同的程序：

```
% f95 -c -o bar.o bar.f
% f95 -c  -o foo.o foo.f
% f95 -o hello_1 bar.o foo.o
```

编译同一个程序并链接到名为 `libexample` 的库中：

```
% f95 -o hello_1 foo.f bar.f -lexample
```

Fortran 文档

有关使用 Fortran 95 编译器以及 `f95` 命令及其选项的说明的完整信息，请参见《Oracle Developer Studio 12.6：Fortran 用户指南》和 `f95`(1) 手册页。有关新功能和已更改功能的信息，请参见《Oracle Developer Studio 12.6 发行版的新增功能》。有关编译器的问题和解决方法以及限制和不兼容性的信息，请参见《Oracle Developer Studio 12.6：发行说明》。

C/C++/Fortran 库

Oracle Developer Studio 编译器利用操作系统的本地库。Oracle Solaris 操作系统提供了许多系统库，这些库安装在 `/usr/lib` 中，包括 C 运行时 `libc` 和多个 C++ 运行时库。`intro`(3) 手册页描述每个库，并引用其他手册页中有关每个库的详细信息。键入 `man intro.3` 命令可以查看该手册页。

要链接 `/usr/lib` 系统库，可以在使用编译器时使用相应的 `-l` 选项。例如，要链接 `libmalloc` 库，可以在链接时在 `cc` 和 `CC` 命令行上指定 `-lmalloc`。

除了操作系统中提供的运行时库以外，还有随 Oracle Developer Studio 提供的 Fortran、C 和 C++ 的运行时库。有些示例包含了 `libsunmath` 和 `libmopt` 数学库。

Fortran 运行时库随 Oracle Developer Studio 提供，而不随操作系统提供。

Fortran 程序还可以使用具有 C 接口的 Oracle Solaris `/usr/lib` 库。有关 C-Fortran 接口的信息，请参见《*Fortran* 编程指南》。

有关链接库的更多信息，请参见 Oracle Solaris 文档中的《链接程序和库指南》。

用于并行编程的 OpenMP 4.0

OpenMP 是一种应用编程接口 (Application Programming Interface, API)，可用 C、C++ 和 Fortran 编写共享内存并行应用程序。该编程接口包含一套编译器指令、库例程和环境变量。

用 OpenMP 编程有以下优势：

- 极大地提高了最新多核体系结构上的程序性能。
- 使程序员可以轻松编写可移植代码，因为有大量的编译器支持 OpenMP。
- 所需编程工作不多。程序员可以确定现有程序中可并行化的代码，然后添加 pragma 将其并行化。
- 支持程序员以递增方式完成代码并行化。

为了充分利用编译器 OpenMP 支持，可以使用 OpenMP 指令和函数来并行化代码部分，并在编译时使用 `-xopenmp` 选项。有关更多信息，请参见《Oracle Developer Studio 12.6：OpenMP API 用户指南》。

适合密集计算程序的 Oracle Developer Studio 性能库

Oracle Developer Studio 性能库是一组优化的高速数学子例程，用于解决线性代数和其他数字密集型问题。Oracle Developer Studio 性能库以来自 `http://www.netlib.org` 上

的 Netlib 的公共域子例程集合为基础。Oracle 增强了这些公共域子例程，并将其捆绑成 Oracle Developer Studio 性能库。

Oracle Developer Studio 性能库例程可以提高多核和多处理器 (multiprocessor, MP) Oracle 系统上的应用程序性能。许多例程进行了基本 Netlib 库中没有的特定于 SPARC 和 x86 的优化，还使用 OpenMP 进行了并行化。除了标准 Fortran 接口以外，还包括了一套完整的 C 接口。

Oracle Developer Studio 性能库通过 `-library` 开关而非用于链接其他库的 `-l` 开关链接到应用程序。

编译使用性能库例程的 Fortran 源代码：

```
% f95 -dalign filename.f -library=sunperf
```

需要使用 `-dalign` 选项，因为该选项用于编译性能库以控制数据对齐。

编译使用性能库例程的 C 或 C++ 源代码：

```
% cc filename.c -library=sunperf
% CC filename.cpp -library=sunperf
```

要静态执行编译和链接以便能够将应用程序部署到未安装 Oracle Developer Studio 性能库的系统中，必须使用 `-library=sunperf` 和 `-staticlib=sunperf` 选项。

有关使用 Oracle Developer Studio 性能库的完整信息，请参见《Oracle Developer Studio 12.6：性能库用户指南》。有关库中每个函数和子例程的手册页，请参见手册页的 3p 部分。有关 Oracle Developer Studio 性能库的新功能和已更改功能的信息，请参见《Oracle Developer Studio 12.6 发行版的新增功能》。

Oracle Developer Studio 代码安全性检查－Discover ADI

`discover -i` ADI 工具（或 `libdiscoverADI.so` 库）是 Oracle Developer Studio 中的一项安全功能，它利用 SPARC M7 硬件轻松修复错误代码。它通过较早地定位错误代码来防止恶意软件攻击及其他内存损坏问题。

为了防止此类内存损坏，`discover` ADI 插入了生产代码，检查分配给应用程序内存指针的版本号或颜色以及与内容的版本号关联的指针。如果指针版本号与内容版本号不匹配，`discover` ADI 会报告错误。此硬件协助式检查允许用户检测缓冲区溢出、释放的指针和过时的指针等错误。`discover` ADI 随后会报告确切的源代码行（使用 `-g` 选项进行编译时）和错误的堆栈跟踪，以便用户可以修复错误代码，然后再继续执行。此外，还会报告错误的分配和自由点堆栈跟踪。无需代码检测或重建步骤。有关更多信息，请参见《Oracle Developer Studio 12.6：Discover 和 Uncover 用户指南》 中的 “Hardware-Assisted Checking Using Silicon Secured Memory (SSM)”。

如果应用程序具有自己的内存分配器并且不调用 malloc()，则用户可以使用 discover ADI 工具提供的新 API。要查看这些新 API，请参见《Oracle Developer Studio 12.6：Discover 和 Uncover 用户指南》 中的 “Custom Memory Allocators and the discover ADI Library”。libdiscoverADI.so 库或 libaidiplugin.3 手册页中也会提供这些 API。

用于生成应用程序的 dmake 实用程序

dmake 实用程序是一个命令行工具，与 make(1) 兼容，用于生成在 makefile 中定义的软件项目目标。dmake 可以采用网格模式、分布模式、并行模式或串行模式生成目标。如果使用的是标准 make(1) 实用程序，在对 makefile 进行任何更改时可以毫不费力地过渡到使用 dmake。dmake 是 make 实用程序的超集。

dmake 解析您的 makefile，以确定哪些目标可以并发生成，并将这些目标的生成作业分配到您在 .dmakerc 文件中指定的一定数目的主机中。

当您使用项目的 makefile 中的目标在 IDE 中生成并运行项目时，缺省情况下，Oracle Developer Studio 将使用 dmake。也可以使用 dmake 通过 IDE 执行各个 makefile 目标。也可以将 IDE 配置为改用 make。

有关如何从命令行使用 dmake 以及如何创建 .dmakerc 文件的信息，请参见《Oracle Developer Studio 12.5：分布式创建 (dmake)》或 dmake(1) 手册页。

用于调试应用程序的工具

Oracle Developer Studio 附带 dbx 调试器，可以帮助您检测应用程序中的错误。

dbx 是一个交互式源代码级命令行调试工具。可以使用它来以可控方式运行 C、C++ 或 Fortran 程序以及检查已停止程序的状态。使用 dbx 可以完全控制程序的动态执行过程，包括收集性能和内存使用情况数据、监视内存访问及检测内存泄漏。

dbx 允许您执行以下任务：

- 检查已崩溃程序的信息转储文件
- 设置断点
- 单步执行程序
- 检查调用堆栈
- 对变量和表达式求值
- 使用运行时检查来发现内存访问问题和内存泄漏
- 使用“修复并继续”来修改和重新编译源文件并继续执行，而不重新生成整个程序

您可以在命令行、以图形方式通过 Oracle Developer Studio IDE 或者通过称为 dbxtool 的单独图形界面来使用 dbx 调试器。

有关在不同用户界面中使用 dbx 的更多信息，请参见以下部分：

- “命令行中的 dbx” [21]
- “IDE 中的 dbx” [22]
- “dbxtool 中的 dbx” [23]

命令行中的 dbx

启动 dbx 的 dbx 命令的基本语法如下：

```
dbx [options] [program-name|-] [process-ID]
```

启动 dbx 会话并装入要调试的程序 test：

```
% dbx test
```

启动 dbx 会话并将其连接到进程 ID 为 832 的正在运行的程序：

```
% dbx - 832
```

当 dbx 会话启动时，dbx 装入您正在调试的程序的信息。然后，dbx 以就绪状态等待访问程序的主程序块，如 C 或 C++ 程序中的 main（） 函数。将显示 (dbx) 命令提示符。

您可以在 (dbx) 提示符下键入命令。一般来说，应先键入 stop in main 这样的命令来设置断点，然后键入 run 命令运行程序：

```
(dbx) stop in main
(4) stop in main
(dbx) run
Running: quote_1
(process id 5685)
(dbx)
```

执行过程在断点处停止时，可以键入 step 和 next 这样的命令单步执行代码，键入 print 和 display 来对表达式和变量求值。

有关 dbx 实用程序的命令行选项的信息，请参见 dbx(1) 手册页。

有关使用 dbx 的完整信息（包括命令参考部分），请参见《Oracle Developer Studio 12.6：使用 dbx 调试程序》。还可以在 (dbx) 命令行键入 help 来了解有关 dbx 命令的信息。

有关新功能和已更改功能的列表，请参见《Oracle Developer Studio 12.6 发行版的新增功能》。

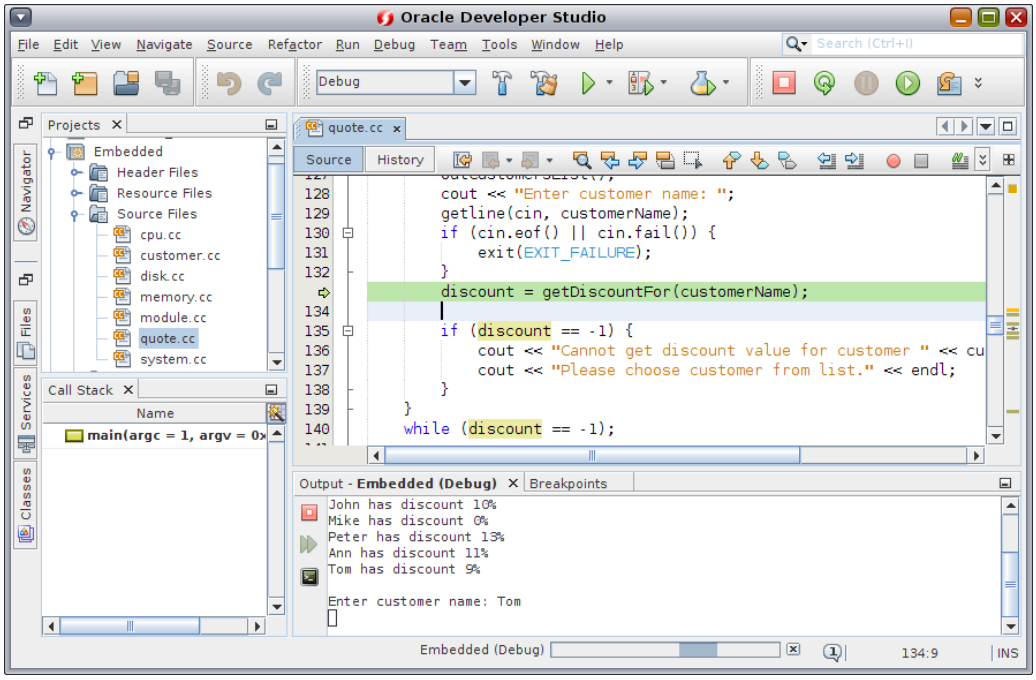
有关当前发行版的 dbx 中的已知问题、限制和不兼容性，请参见《Oracle Developer Studio 12.6：发行说明》。

IDE 中的 dbx

您可以在 Oracle Developer Studio IDE 中使用 dbx，方法为打开项目，在源代码中创建断点，然后单击 "Debug"（调试）按钮。IDE 允许您使用菜单选项和按钮来分步执行程序，并提供一组完整的调试窗口。

与生成应用程序一样，IDE 将应用程序作为一个项目来调试。此外，还可以使用 IDE 调试不与 IDE 项目关联的可执行文件。

在下面的屏幕抓图中，dbx 中正运行一个 IDE 样例项目。您可以使用 "Debug"（调试）菜单中的命令或 IDE 窗口右上部的按钮来控制调试器。使用 "Debug"（调试）命令和按钮时，IDE 向 dbx 发出命令，并在各个调试窗口中显示输出。



在图中，调试器在一个断点处停止，"Output"（输出）窗口显示程序交互。有些调试器窗口（如 "Variables"（变量）和 "Breakpoints "（断点）窗口）也显示出来，但并未被选中。您可以从 "Window"（窗口）-> "Debugging"（调试）菜单中选择，以打开更多调试窗口。其中一个调试窗口是 "Debugger Console"（调试器控制台）窗口，其中显示与 dbx 的交互。还可以在 "Debugger Console"（调试器控制台）窗口的 (dbx) 提示符下键入命令。

有关在 IDE 中使用 dbx 的更多信息，请参见 IDE 中的集成帮助和《Oracle Developer Studio 12.6：IDE 快速入门教程》。

dbxtool 中的 dbx

您还可以通过 dbxtool 来使用 dbx，该工具是一个独立于 IDE 但具有相似的调试窗口和编辑器的图形工具。与 IDE 不同的是，dbxtool 不使用项目，您可以使用它来调试任何 C、C++ 或 Fortran 可执行文件或信息转储文件。

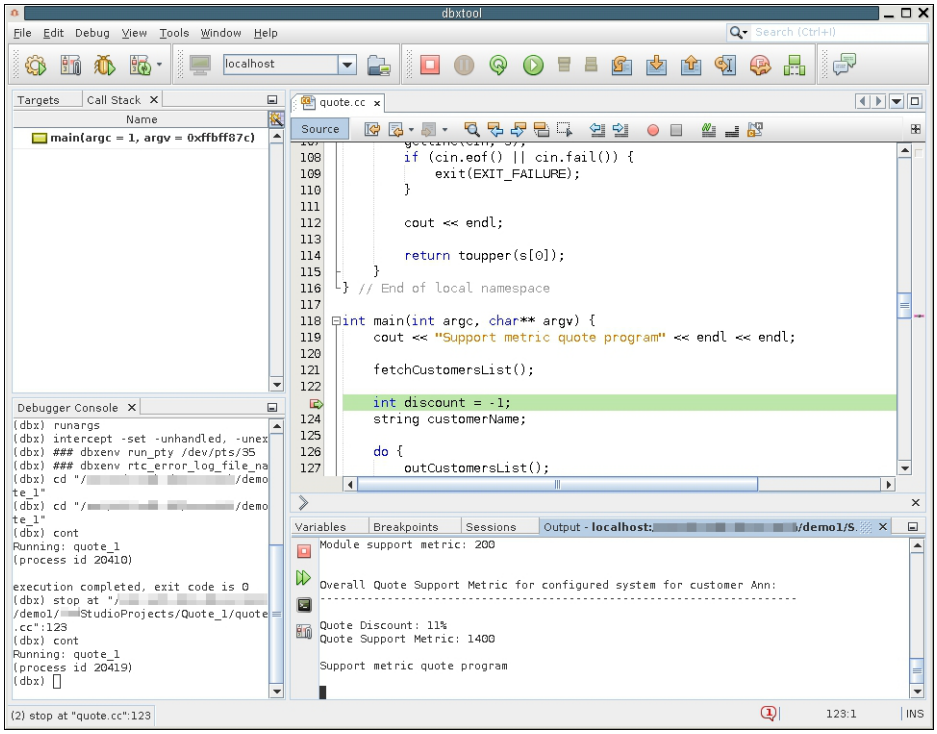
要启动 dbxtool，请键入：

```
% dbxtool executable-name
```

此外，还可以忽略可执行文件名称，而从 dbxtool 内指定。

同 IDE 一样，您可以在 dbxtool 中单击工具栏按钮或使用 "Debug"（调试）菜单选项来向 dbx 发出命令。还可以在 "Debugger Console"（调试器控制台）窗口的 (dbx) 提示符下键入命令。

在下图中，正在 dbxtool 中对 quote_1 程序运行 dbx。已选择 "Debugger Console"（调试器控制台）窗口，您可以看到 (dbx) 提示符以及为响应用户选择由 dbxtool 所输入的命令。



有关使用 dbxtool 的信息，请参见 dbxtool(1) 手册页以及 dbxtool 中的集成帮助。《Oracle Developer Studio 12.6：dbxtool 教程》中介绍了如何使用 dbxtool。

用于验证应用程序的工具

Oracle Developer Studio 提供了一些工具，可以帮助您验证应用程序的稳定性。以下工具结合了动态、静态和代码覆盖分析，以检测应用程序漏洞，包括内存泄漏和内存访问违规。

discover	一种命令行实用程序，可以帮助检测代码中的内存访问错误。
uncover	一种命令行实用程序，显示应用程序代码的哪些区域不在测试所涵盖的范围内。
代码分析器	一种图形工具，用来分析并显示 C 或 C++ 编译器所收集的静态代码错误数据，以及 discover 和 uncover 收集的数据。代码分析器将静态错误数据与动态内存访问错误数据和代码覆盖数据相结合，从而使您能够在应用程序中发现单独使用其他错误检测工具无法找到的错误。

codean　　一种命令行实用程序，提供类似于代码分析器的功能。

用于检测内存错误的 discover 工具

内存错误搜索工具 (discover) 是用于检测程序中内存访问错误的高级开发工具。使用 -g 选项编译二进制文件使 discover 可以在报告错误和警告时显示源代码和行号信息。

discover 实用程序的用法很简单。在使用 -g 选项编译二进制文件后，对二进制文件运行 discover 命令来对其进行检测。然后，运行经检测的二进制文件，以创建 discover 报告。您可以请求 HTML 格式、文本格式或者这两种格式的 discover 报告。该报告显示内存错误、警告和内存泄漏，您可以显示每个错误或警告所对应的源代码和堆栈跟踪。

以下示例来自 discover(1) 手册页，该示例说明了如何准备、检测及运行可执行文件，以生成一份 discover 报告，用于检测内存访问错误。discover 命令行上的 -w 选项指示报告应采用文本格式，-o 选项则指示应输出到屏幕上。

```
% cc -g -O2 test.c -o test.prep
% discover -w - -o test.disc test.prep
% ./test.disc
ERROR (UMR): accessing uninitialized data from address 0x5000c (4 bytes) at:
     foo() + 0xdc  <ui.c:6>
          3:    int *t;
          4:    foo() {
          5:     t = malloc(5*sizeof(int));
          6:=>   printf("%d0, t[1]);
          7:    }
          8:
          9:    main()
     main() + 0x1c
     _start() + 0x108
    block at 0x50008 (20 bytes long) was allocated at:
     malloc() + 0x260
     foo() + 0x24  <ui.c:5>
          2:
          3:    int *t;
          4:    foo() {
          5:=>   t = malloc(5*sizeof(int));
          6:     printf("%d0, t[1]);
          7:    }
          8:
     main() + 0x1c
     _start() + 0x108

     ***************** Discover Memory Report *****************

1 block at 1 location left allocated on heap with a total size of 20 bytes

    1 block with total size of 20 bytes
     malloc() + 0x260
     foo() + 0x24  <ui.c:5>
          2:
          3:    int *t;
          4:    foo() {
          5:=>   t = malloc(5*sizeof(int));
```

```
     6:     printf("%d0, t[1]);
     7:    }
     8:
main() + 0x1c
_start() + 0x108
```

有关更多信息，请参见 discover(1) 手册页和《Oracle Developer Studio 12.6：Discover 和 Uncover 用户指南》。

用于度量代码覆盖的 uncover 工具

uncover 是一种用于度量代码覆盖的命令行工具。该工具显示当应用程序运行时，已实施、未实施以及未涵盖在测试范围内的代码区域。Uncover 会生成一份报告，其中包含统计信息和各项度量，可以帮助您确定应在测试套件中添加哪些函数，以确保测试期间覆盖更多代码。

uncover 可以与使用 Oracle Developer Studio 编译器所生成的任何二进制文件结合使用，而当生成不包含优化的二进制文件时效果最佳。使用 -g 选项编译二进制文件使 uncover 可以在报告代码覆盖情况时显示源代码和行号信息。

在编译了二进制文件后，对二进制文件运行 uncover 命令。uncover 将创建添加了检测代码的新的二进制文件，还将创建名为 *binary*.uc 的目录，其中将包含程序的代码覆盖数据。每次运行已检测的二进制文件时，都将收集代码覆盖数据并存储在 *binary*.uc 目录中。

您可以在性能分析器中显示实验数据，或者生成 HTML 格式的 uncover 报告并用 Web 浏览器显示。

下例说明如何准备、检测及运行可执行文件，以生成用于检查代码覆盖的 uncover 报告。已优化二进制文件是 test，由名称也为 test 的已检测二进制文件所取代。

```
% cc -g -O2 test.c -o test
% uncover test
% test
```

实验目录是 test.uc，其中包含已检测的 test 运行时所生成的数据。test.uc 目录还包含未检测的 test 二进制文件的副本。

在性能分析器中查看实验：

```
% uncover test.uc
```

在浏览器的 HTML 页面查看实验：

```
% uncover -H test.html test.uc
```

有关更多信息，请参见 uncover(1) 手册页和《Oracle Developer Studio 12.6：Discover 和 Uncover 用户指南》。

用于集成错误检查的代码分析器工具

Oracle Developer Studio 代码分析器是一种图形工具，可用来对代码执行集成分析。代码分析器使用您通过其他工具收集的三种类型的信息：

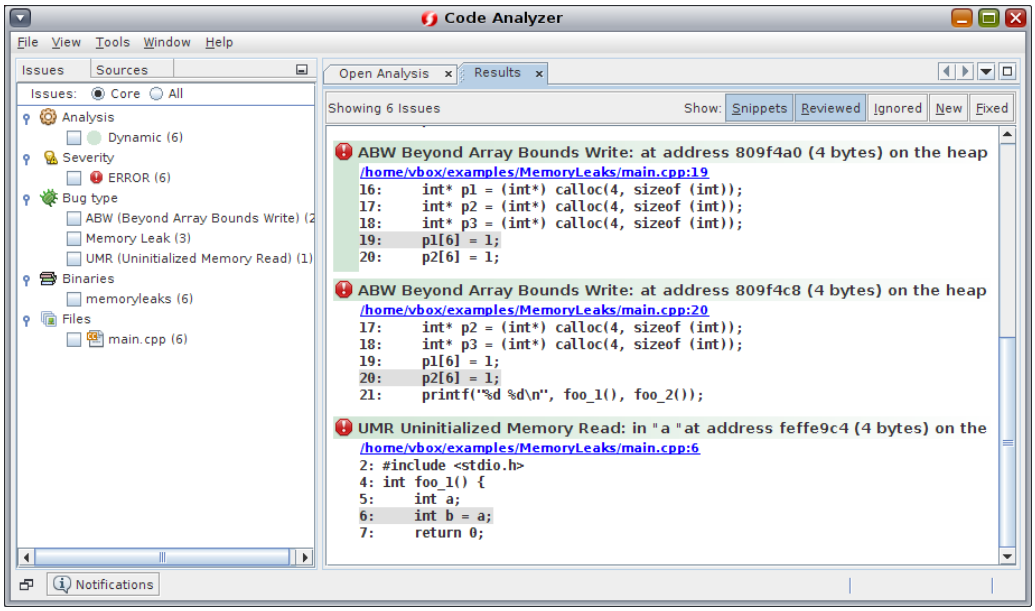
- 静态代码检查，当您使用 Oracle Developer Studio C 或 C++ 编译器并指定 `-xprevise=yes` 选项来编译应用程序时，执行该检查。
- 动态内存访问检查，当您使用带有 -a 选项的 `discover` 来检测二进制文件，然后运行检测后的二进制文件时，执行该检查。
- 代码覆盖检查，当您使用 `uncover` 检测二进制文件，运行检测后的二进制文件，然后对收集的覆盖数据运行带有 -a 选项的 Uncover 时，执行该检查。

您可以对已使用其中的任一种工具或其任意组合来准备的二进制文件使用代码分析器。但是，综合查看这三种类型的数据可以最为详细地了解您的代码，因而使您能够创建更安全、更可靠的应用程序。

下例说明了如何对已经使用 `discover` 和 `uncover` 准备的二进制文件 `a.out` 运行代码分析器。

```
% code-analyzer a.out
```

在下图中，代码分析器正在显示在二进制文件 `a.out` 中发现的问题。



有关更多信息，请参见集成的代码分析器帮助、《Oracle Developer Studio 12.6：代码分析器用户指南》和《Oracle Developer Studio 12.6：代码分析器教程》。

用于集成检查的 codean 工具

此外，还可以使用 codean 命令行实用程序，根据通过编译器、discover 和 uncover 收集的数据生成报告。codean 工具提供类似于代码分析器的功能，但可在没有图形环境的系统上使用，也可供偏好命令行的用户使用。codean 工具还可用于自动化脚本，并且有一些在代码分析器工具中尚未提供的功能。

有关更多信息，请参见 codean(1) 手册页、《Oracle Developer Studio 12.6：代码分析器用户指南》和《Oracle Developer Studio 12.6：代码分析器教程》。

用于调整应用程序性能的工具

Oracle Developer Studio 软件包括了多个可用于检查应用程序的行为并调整其性能的工具。

这些性能工具包括：

- **性能分析器和关联的工具。**一组高级性能工具和实用程序，用于帮助您发现代码中影响性能的地方。
- **IDE 中的分析工具。**可用于从 IDE 内部检查项目的性能。

性能分析器工具

Oracle Developer Studio 软件提供了一组可以结合使用的高级性能工具和实用程序。收集器、性能分析器、线程分析器和 er_print 实用程序有助于您评估代码的性能、识别潜在性能问题并定位出现这些问题的代码部分。这些工具统称为性能分析器工具。

可以选择使用 Oracle Developer Studio C、C++ 和 Fortran 编译器，以实现用于提高程序性能的硬件和高级优化技术。同时，性能分析器工具也设计为在 Oracle Sun 硬件上与编译器一起使用，在 Oracle Sun 计算机上运行时可帮助您提高程序的性能。

利用性能分析器工具，可以控制收集的数据、深入检测数据，以及检查程序与硬件的交互。性能分析器工具设计用于在最新 Oracle Sun 硬件上运行的复杂的、计算密集型应用程序，并经过此类测试。

性能分析器工具还能够对 OpenMP 并行应用程序和基于 MPI 的分布式应用程序进行分析，帮助您确定是否在应用程序中有效地使用了这些技术。

要使用性能分析器工具，必须执行以下两步：

1. 在性能分析器中分析目标应用程序，或者使用 `collect` 命令从目标应用程序收集性能数据。
2. 使用性能分析器图形工具、`er_print` 命令行实用程序或线程分析器图形工具检查数据，检查是否存在数据争用和死锁数据。

收集性能数据以分析应用程序

收集器通过使用分析和跟踪函数调用来收集性能数据。这些数据可能包括调用堆栈、微状态计数信息（仅在 Oracle Solaris 平台上）、线程同步延迟数据、硬件计数器溢出数据、消息传递接口 (Message Passing Interface, MPI) 函数调用数据、内存分配数据以及操作系统和进程的摘要信息。收集器可收集 C、C++ 和 Fortran 程序的所有类型的数据，以及用 Java 编程语言编写的应用程序的分析数据。可使用 `collect` 命令、从性能分析器中的 "Profile Application"（分析应用程序）对话框或使用 `dbx` 调试器的 `collect` 子命令来运行收集器。

Oracle Developer Studio IDE 分析工具也使用收集器来收集信息。

要使用 `collect` 命令收集数据：

```
% collect [collect-options] executable executable-options
```

可向 `collect` 命令添加选项来指定想要收集的数据类型。例如，使用 `-i on` 选项时，收集器将执行输入/输出跟踪。通过在可执行文件之后指定参数，可将参数传递到目标可执行文件。

缺省情况下，收集器会创建一个名为 `test.1.er` 的数据目录，但可在命令行上指定其他名称。`test.1.er` 目录为实验目录，其名称必须始终以 `.er` 结尾，以便工具将其识别为实验目录。

以下命令显示如何在 `synprog` 程序上使用 `collect`：

```
% collect synprog

Creating experiment database test.1.er (Process ID: 11103) ...
00:00:00.000  ===== (11103) synprog run
00:00:00.005  ===== (11103) Mon  22 Sep 14  17:05:51 Stopwatch calibration
  OS release 5.11 -- enabling microstate accounting 5.11.
        0.000096 s.  (22.4 % of 0.000426 s.) -- inner
 N = 1000, avg = 0.096 us., min = 0.090, max = 0.105
        0.000312 s.  (67.0 % of 0.000466 s.) -- outer
 N = 1000, avg = 0.312 us., min = 0.307, max = 0.457
00:00:00.006  ===== (11103)  Begin commandline
 icpu.md.cpu.rec.recd.dousl.gpf.fitos.uf.ec.tco.b.nap.sig.sys.so.sx.so
00:00:00.006  ===== (11103) start of icputime
    3.003069 wall-secs.,   2.978360 CPU-secs., in icputime
00:00:03.009  ===== (11103) start of muldiv
    3.007489 wall-secs.,   2.997647 CPU-secs., in muldiv
00:00:06.017  ===== (11103) start of cputime
```

```
    3.002315 wall-secs.,   2.989407 CPU-secs., in cputime
00:00:09.019  ===== (11103) start of recurse
    3.082371 wall-secs.,   3.069782 CPU-secs., in recurse
    …
     （为了节省空间而对输出进行了编辑）
    …
```

数据存储在 `test.1.er` 目录中，可使用性能分析器或 `er_print` 进行查看。

有关对可以下载的样例应用程序使用性能分析器的逐步说明，请参见《Oracle Developer Studio 12.6：性能分析器教程》。

有关如何分析应用程序以及使用收集器的详细信息，请参见性能分析器中的 "Help"（帮助）菜单、《Oracle Developer Studio 12.6：性能分析器》手册以及 `collect`(1) 手册页。

使用性能分析器检查性能数据

利用性能分析器可以深入了解应用程序的行为，从而能够找出代码中的问题方面。性能分析器可确定哪些函数、代码段和源代码行占用的系统资源最多。性能分析器可以分析单线程、多线程和多进程应用程序，然后提供分析数据以帮助确定可提高应用程序性能的方面。

可使用 `analyzer` 命令运行性能分析器。启动性能分析器的 `analyzer` 命令的基本语法如下：

```
% analyzer [experiment-list]
```

experiment-list 是使用收集器收集的实验的一个或多个文件名。如果想要装入多个实验，请以空格为分隔符来指定名称。针对多个实验调用性能分析器时，默认情况下，它会对实验数据进行聚集。但是，还可以通过在命令行上在实验名称之前指定 -c 选项来对实验进行比较。

如果不在命令行上指定实验，则性能分析器将显示 "Welcome"（欢迎）屏幕以帮助您开始。

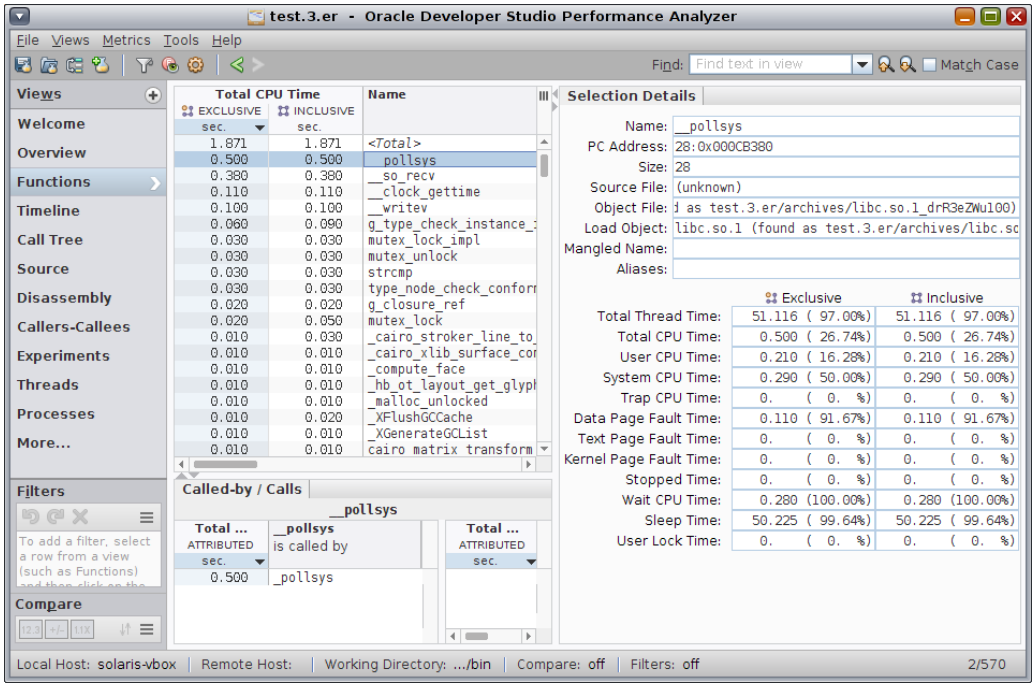
要在性能分析器中打开实验 `test.1.er`：

```
% analyzer test.1.er
```

实验的初始视图是 "Overview"（概述），可在其中快速了解程序所用时间和资源的概况，并可选择要在性能数据视图中看到的性能度量。

下图显示性能分析器 "Functions"（函数）视图，对 `synprog` 示例进行 `test.1.er` 实验。"Functions"（函数）视图显示由 `synprog` 程序的各个函数所使用的 CPU 时间。单击 `gpf_work` 函数时，右侧的 "Selection Details"（选择详细信息）窗口显示 `gpf_work` 函数资源使用情况的详细信息。在 "Functions"（函数）视图底部，"Called-by/Calls"（调用方

/调用）区域显示被 gpf_work 调用的函数；双击调用可导航到 "Functions"（函数）视图中的相应的函数。



有关使用性能分析器的信息，请参见《Oracle Developer Studio 12.6：性能分析器》手册、性能分析器集成帮助以及 analyzer(1) 手册页。

有关对可以下载的样例应用程序使用性能分析器的逐步说明，请参见《Oracle Developer Studio 12.6：性能分析器教程》。

使用 er_print 实用程序检查性能数据

er_print 实用程序以纯文本格式显示性能分析器中呈现的大多数显示内容，但时间线显示、MPI 时间线显示和 MPI 图表显示除外。

可以使用 er_print 实用程序显示函数、调用方和被调用方的性能度量、调用树、源代码列表、反汇编代码列表、抽样信息、数据空间数据、线程分析数据以及执行统计信息。

er_print 命令的一般语法如下：

```
% er_print -command experiment-list
```

可以指定一个或多个命令以指明要显示的数据的类型。*experiment-list* 是使用收集器收集的实验的一个或多个文件名。对多个实验进行调用时，缺省情况下，`er_print` 将聚集实验数据，但也可用于对实验进行比较。

以下示例说明了用于显示程序函数信息的命令。所示输出针对在本文档上一节的性能分析器屏幕抓图中使用的同一个实验。

```
%  er_print -functions test.1.er
Functions sorted by metric: Exclusive Total CPU Time

Excl.     Incl.      Name
Total     Total
CPU sec.  CPU sec.
50.806    50.806     <Total>
 5.994     5.994     so_burncpu
 5.914     5.914     real_recurse
 3.502     3.502     gpf_work
 3.012     3.012     sigtime_handler
 3.002     3.002     bounce_a
 3.002     3.002     cputime
 3.002     3.002     icputime
 2.992     2.992     sx_burncpu
 2.992     2.992     underflow
 2.792     2.792     muldiv
 2.532     2.532     my_irand
 1.831     1.831     gethrtime
 1.031     1.991     tailcall_b
 0.961     0.961     inc_middle
 0.961     0.961     tailcall_c
 0.941     0.941     gethrvtime
 0.941     0.941     gettimeofday
 0.911     2.902     tailcall_a
 0.801     0.801     dousleep
 0.650     0.650     inc_entry
 0.640     0.640     inc_exit
 0.480     3.012     fitos
 0.330     0.330     inc_func
 0.320     0.320     inc_body
 0.320     0.320     inc_brace
 0.290     4.003     systime
 0.260     0.260     ext_macro_code
```

删除了若干行

如果在启动 `er_print` 时指定实验名称并忽略命令，还可以交互方式使用 `er_print`。您可以在 `(er_print)` 提示符下键入命令。

有关 `er_print` 实用程序的信息，请参见《Oracle Developer Studio 12.6：性能分析器》手册和 `er_print`(1) 手册页。

使用线程分析器分析多线程应用程序性能

线程分析器是性能分析器的一个专门版本，用于检查多线程程序。线程分析器可以检测导致用 POSIX 线程 API、Oracle Solaris 线程 API、OpenMP 指令或混用这几项编写的代码中出现数据争用和死锁的多线程编程错误。

线程分析器用于检测多线程程序中的两个常见线程问题：

- 数据争用问题，在单个进程中的两个线程同时访问同一个共享内存位置且没有独占锁定同时至少一个访问为写访问时发生。
- 死锁问题，当两个或多个线程由于相互等待另一线程完成任务而被阻塞时发生。

线程分析器已针对多线程程序分析进行了简化，仅显示性能分析器的 "Races"（争用）、"Deadlocks"（死锁）和 "Dual Source"（双源）数据视图。对于 OpenMP 程序，还显示 "OpenMP Parallel Region"（OpenMP 并行区域）和 "OpenMP Task"（OpenMP 任务）视图。

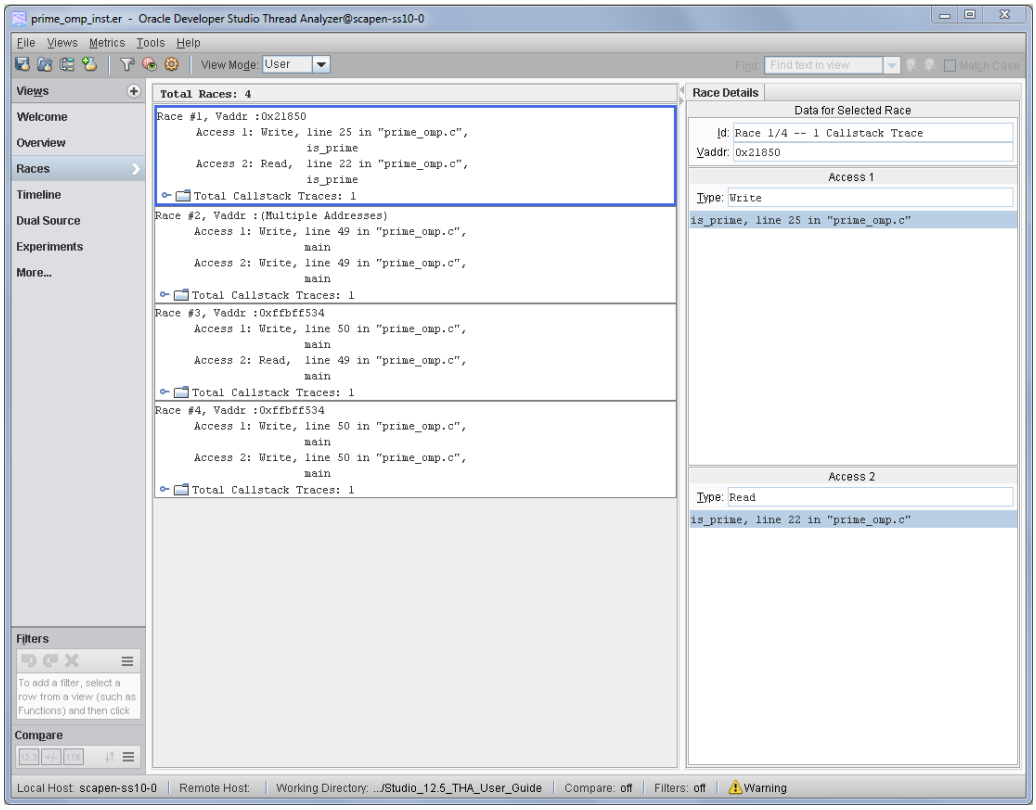
您可以检测源代码或二进制代码中的数据争用。在这两种情况下，您必须检测代码才能收集到必要的数据。

使用线程分析器：

1. 检测代码以分析数据争用。对于源代码，请在编译时使用 `-xinstrument=datarace` 编译器选项。对于二进制代码，使用 `discover -i datarace` 命令创建检测后的二进制文件。

   死锁检测不需要测试。
2. 将 `collect` 命令与 `-r race` 选项连用来运行可执行文件以收集数据争用数据，与 `-r deadlock` 选项连用来收集死锁数据或者与 `-r all` 选项连用来收集这两种类型的数据。
3. 使用 `tha` 命令启动线程分析器，或使用 `er_print` 命令显示生成的实验。

下图显示的线程分析器窗口包含在 OpenMP 程序中检测到的数据争用以及造成数据争用的调用堆栈。



有关使用线程分析器的信息，请参见 tha(1) 手册页和《Oracle Developer Studio 12.6：线程分析器用户指南》。

IDE 中的分析工具

Oracle Developer Studio IDE 提供的交互式图形分析工具可用于检查在 IDE 内部运行的项目的性能。分析工具使用 Oracle Developer Studio 实用程序和操作系统实用程序来收集数据。

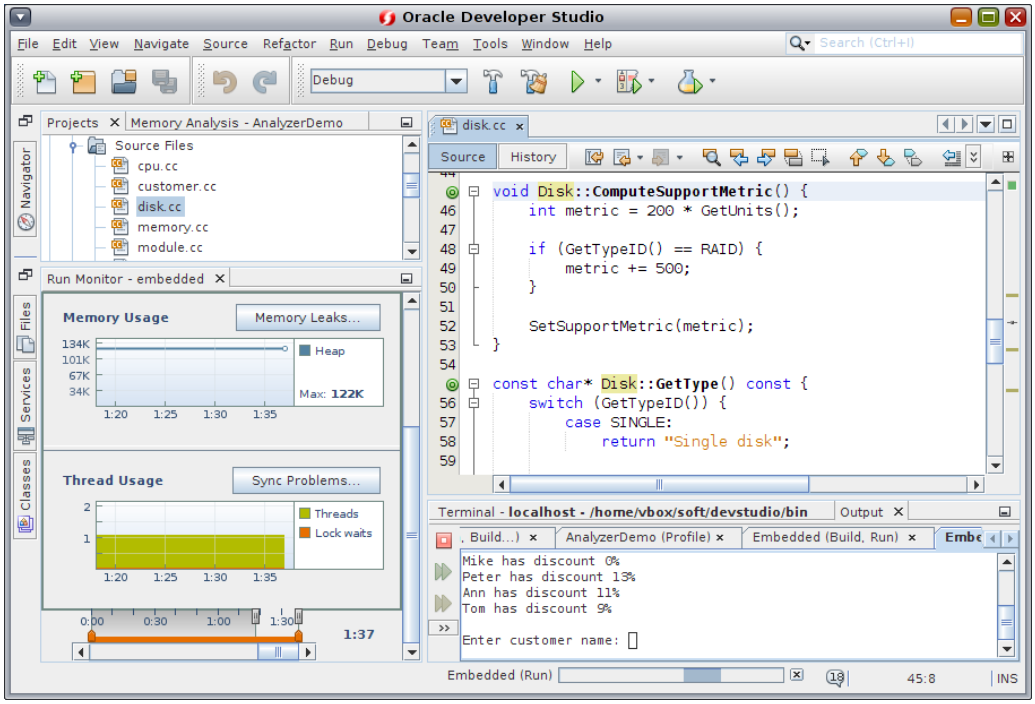
可通过 "Profile Project"（分析项目）按钮  使用分析工具。

Monitor Project（监视项目）	提供一些图形，可籍此了解程序的资源使用情况摘要。

Memory Access Errors（内存访问错误）	分析运行中的程序以检测内存访问错误和内存泄漏。
Data Races and Deadlocks Detection（数据争用和死锁检测）	分析运行中的程序以检测线程间实际和潜在的数据争用和死锁。

在分析项目并选择监视项目时，将打开 "Run Monitor"（运行监视器）窗口以显示低影响工具的 CPU 使用情况、内存使用情况和线程使用情况的输出。

下图显示使用运行监视器工具的 IDE。

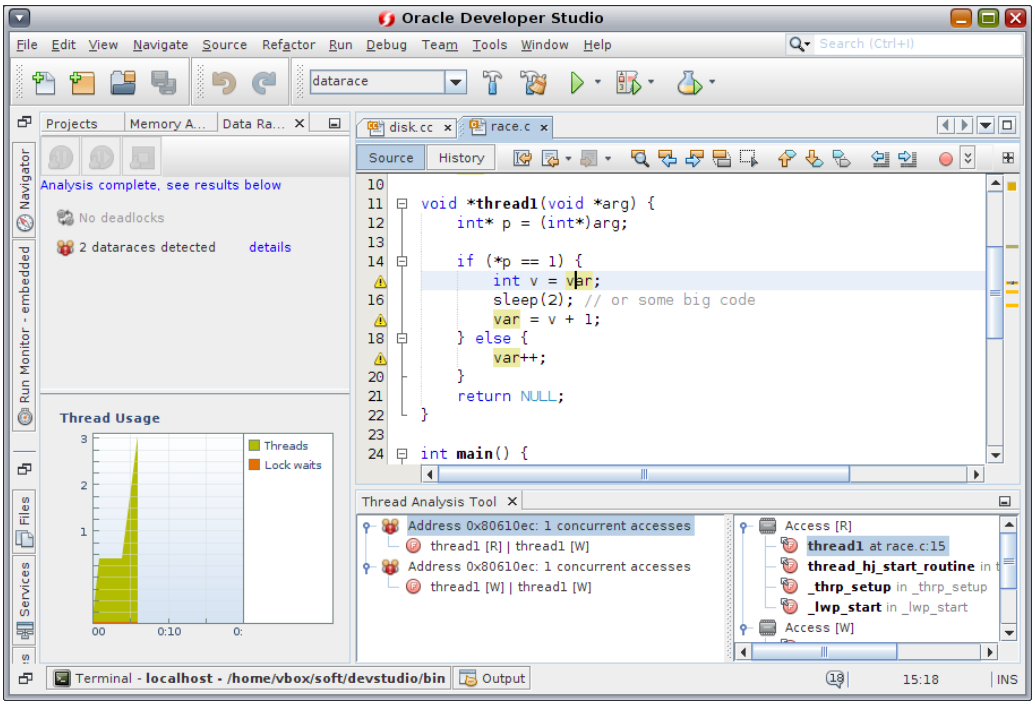


用于执行更详细分析的其他工具对系统和应用程序性能的影响较大，因此运行监视器项目时系统不会自动运行这些工具。高级工具与 "Run Monitor"（运行监视器）工具关联，可通过单击按钮轻松启动以查看 "Hot Spots"（热点）、"Memory Leaks"（内存泄漏）和 "Sync Problems"（同步问题）。

"Data Races and Deadlocks Detection"（数据争用和死锁检测）工具使用的底层技术与线程分析器相同，稍后将在本文档中介绍。该工具将分析添加到线程程序中，然后在程序

运行时分析程序，以检测线程中实际和潜在的数据争用和死锁。要启动该工具，请单击 "Profile Project"（分析项目）按钮，选择 "Data Races and/or Deadlocks"（数据争用和/或死锁），指定用于数据收集的选项，然后单击 "Start"（启动）。

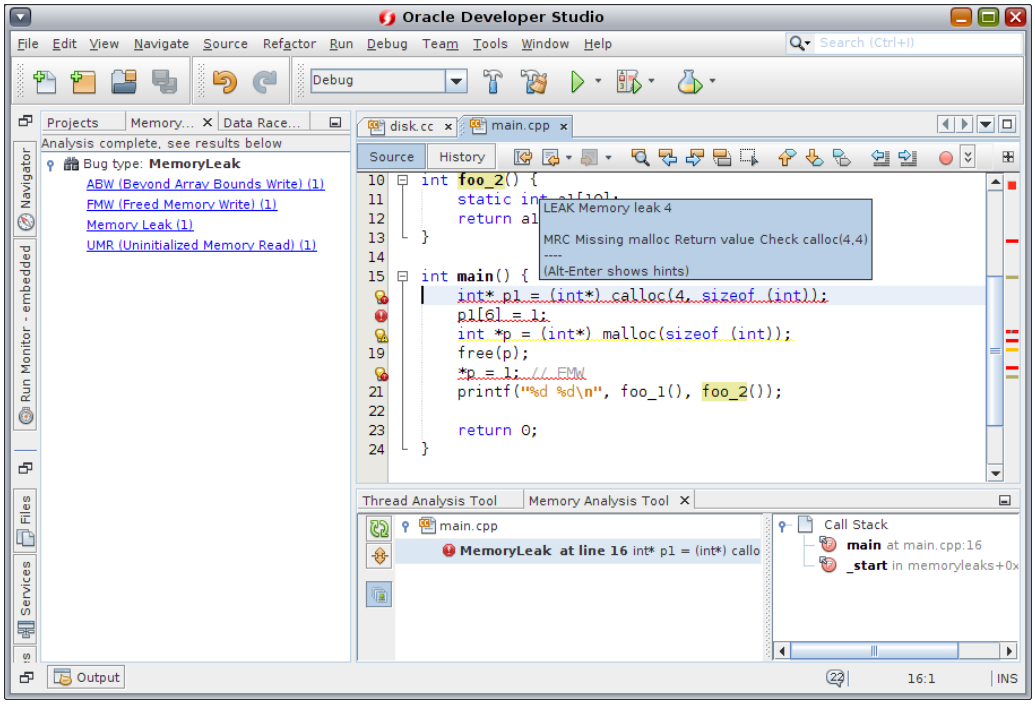
下图显示在检测到数据争用之后的 "Data Races and Deadlocks Detection"（数据争用和死锁检测）工具。



如果在 "Data Race Detection"（数据争用检测）窗口中单击 "details"（详细信息）链接，将打开 "Thread Details"（线程详细信息）窗口以显示发生数据争用的位置。您可以在 "Thread Details"（线程详细信息）窗口中双击线程，以打开发生问题的源文件并转到受影响的代码行。

"Memory Access Error"（内存访问错误）工具使用的底层技术与先前介绍的 discover 相同。该工具会检测程序，然后在程序运行时执行分析以检测内存访问错误和内存泄漏。要启动该工具，请单击 "Profile Project"（分析项目）按钮，选择 "Memory Access Error"（内存访问错误），指定用于数据收集的选项，然后单击 "Start"（启动）。内存访问错误类型将显示在 "Memory Analysis"（内存分析）窗口中。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，在其中可查看每个错误的调用堆栈。

下图显示在检测到内存访问错误之后的 "Memory Access Error"（内存访问错误）工具。



有关如何使用分析工具的信息，请参见 IDE 集成帮助，访问方法为在 IDE 中按 F1 键或通过 "Help"（帮助）菜单。在 "Help"（帮助）的 "Contents"（内容）标签中参见 "Profiling C/C++/Fortran Applications"（分析 C/C++/Fortran 应用程序）、"Detecting Data Races and Deadlocks"（检测数据争用和死锁）和 "Finding Memory Access Errors in Your Project"（在项目中查找内存访问错误）。

有关更多信息

有关更多信息，请参见 Oracle 技术网上的 Oracle Developer Studio 产品页面。您可以找到白皮书、技术文章、培训和支持信息以及社区论坛和博客，以便更好地利用 Oracle Developer Studio。