# Oracle® Solaris Studio 12.3 概述

文件号码 E26453
2011 年 12 月，E26453-01

ORACLE

120220@25097

# 目录

# 前言

本手册面向首次使用 Oracle Solaris Studio 并想要了解该产品中提供的众多工具、编译器和编程库的 C、C++ 和 Fortran 应用程序开发者。本手册不提供有关如何使用工具的详细信息，但介绍了开发者如何在开发过程中使用这些工具来编辑、生成和分析软件应用程序。

## 支持的平台

本 Oracle Solaris Studio 发行版支持使用 SPARC 系列处理器体系结构的平台（运行 Oracle Solaris 操作系统），以及使用 x86 系列处理器体系结构的平台（运行 Oracle Solaris 或特定的 Linux 系统）。

本文档使用以下术语说明 x86 平台之间的区别：

- "x86" 泛指 64 位和 32 位的 x86 兼容产品系列。
- "x64" 指特定的 64 位 x86 兼容 CPU。
- “32 位 x86”指出了有关基于 x86 的系统的特定 32 位信息。

在 SPARC 和 x86 系统中，特定于 Linux 系统的信息仅指受支持的 Linux x86 平台，而特定于 Oracle Solaris 系统的信息仅指受支持的 Oracle Solaris 平台。

有关支持的硬件平台和操作系统发行版的完整列表，请参见《Oracle Solaris Studio 12.3 发行说明》。

## Oracle Solaris Studio 文档

可以查找 Oracle Solaris Studio 软件的完整文档，如下所述：

- 产品文档位于 Oracle Solaris Studio 文档 Web 站点，包括发行说明、参考手册、用户指南和教程。
- 代码分析器、性能分析器、线程分析器、dbxtool、DLight 和 IDE 的联机帮助可以在这些工具中通过 "Help"（帮助）菜单以及 F1 键和许多窗口和对话框上的 "Help"（帮助）按钮获取。
- 命令行工具的手册页介绍了工具的命令选项。

# 相关的第三方 Web 站点引用

本文档引用了第三方 URL，以用于提供其他相关信息。

---

**注 –** Oracle 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他资料，Oracle 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的或连带产生的实际或名义损坏或损失，Oracle 概不负责，也不承担任何责任。

---

# 开发者资源

对于使用 Oracle Solaris Studio 的开发者，可访问 Oracle 技术网 Web 站点来查找以下资源：

- 有关编程技术和最佳做法的文章
- 软件最新发行版完整文档的链接
- 有关支持级别的信息
- 用户论坛。

# 获取 Oracle 支持

Oracle 客户可通过 My Oracle Support 获取电子支持。有关信息，请访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info`，或访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs`（如果您听力受损）。

# 印刷约定

下表介绍了本书中的印刷约定。

**表 P–1** 印刷约定

| 字体或符号 | 含义 | 示例 |
|---|---|---|
| `AaBbCc123` | 命令、文件和目录的名称；计算机屏幕输出 | 编辑 `.login` 文件。<br>使用 `ls -a` 列出所有文件。<br>`machine_name% you have mail.` |
| **`AaBbCc123`** | 用户键入的内容，与计算机屏幕输出的显示不同 | `machine_name%` **`su`**<br>`Password:` |

表 P–1 印刷约定 （续）

| 字体或符号 | 含义 | 示例 |
| --- | --- | --- |
| *aabbcc123* | 要使用实名或值替换的命令行占位符 | 删除文件的命令为 `rm` *filename*。 |
| *AaBbCc123* | 保留未译的新词或术语以及要强调的词 | 这些称为 *Class* 选项。<br>**注意**：有些强调的项目在联机时以粗体显示。 |
| **新词术语强调** | 新词或术语以及要强调的词 | **高速缓存**是存储在本地的副本。<br>请**勿**保存文件。 |
| 《书名》 | 书名 | 阅读《用户指南》的第 6 章。 |

# 命令中的 shell 提示符示例

下表显示了 Oracle Solaris OS 中包含的缺省 UNIX shell 系统提示符和超级用户提示符。请注意，在命令示例中显示的缺省系统提示符可能会有所不同，具体取决于 Oracle Solaris 发行版。

表 P–2 shell 提示符

| shell | 提示符 |
| --- | --- |
| Bash shell、Korn shell 和 Bourne shell | `$` |
| Bash shell、Korn shell 和 Bourne shell 超级用户 | `#` |
| C shell | `machine_name%` |
| C shell 超级用户 | `machine_name#` |

# Oracle Solaris Studio 12.3 概述

Oracle Solaris Studio 软件是适合 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 处理器的多核系统。

## Oracle Solaris Studio 软件简介

Oracle Solaris 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。

当您开发在 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，或者是开发在 x86 和 x64 平台上的 Oracle Linux 中运行的 C、C++ 和 Fortran 应用程序时，Oracle Solaris Studio 能提供您所需的任何内容。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。

特别是，Oracle Solaris Studio 编译器和分析工具被设计为利用多核 CPU 的处理能力，其中包括 SPARC T4、SPARC T3、UltraSPARC T2 和 UltraSPARC T2 Plus 处理器以及 Intel® Xeon® 和 AMD Opteron 处理器。利用 Oracle Solaris Studio 可以更方便地创建面向这些平台的串行和并行软件应用程序。

Oracle Solaris Studio 的组件包括：

- IDE，用于在图形环境中开发应用程序。Oracle Solaris Studio IDE 集成了多种其他 Oracle Solaris Studio 工具，采用 Oracle Solaris 技术，如 DTrace。
- 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 Solaris 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。

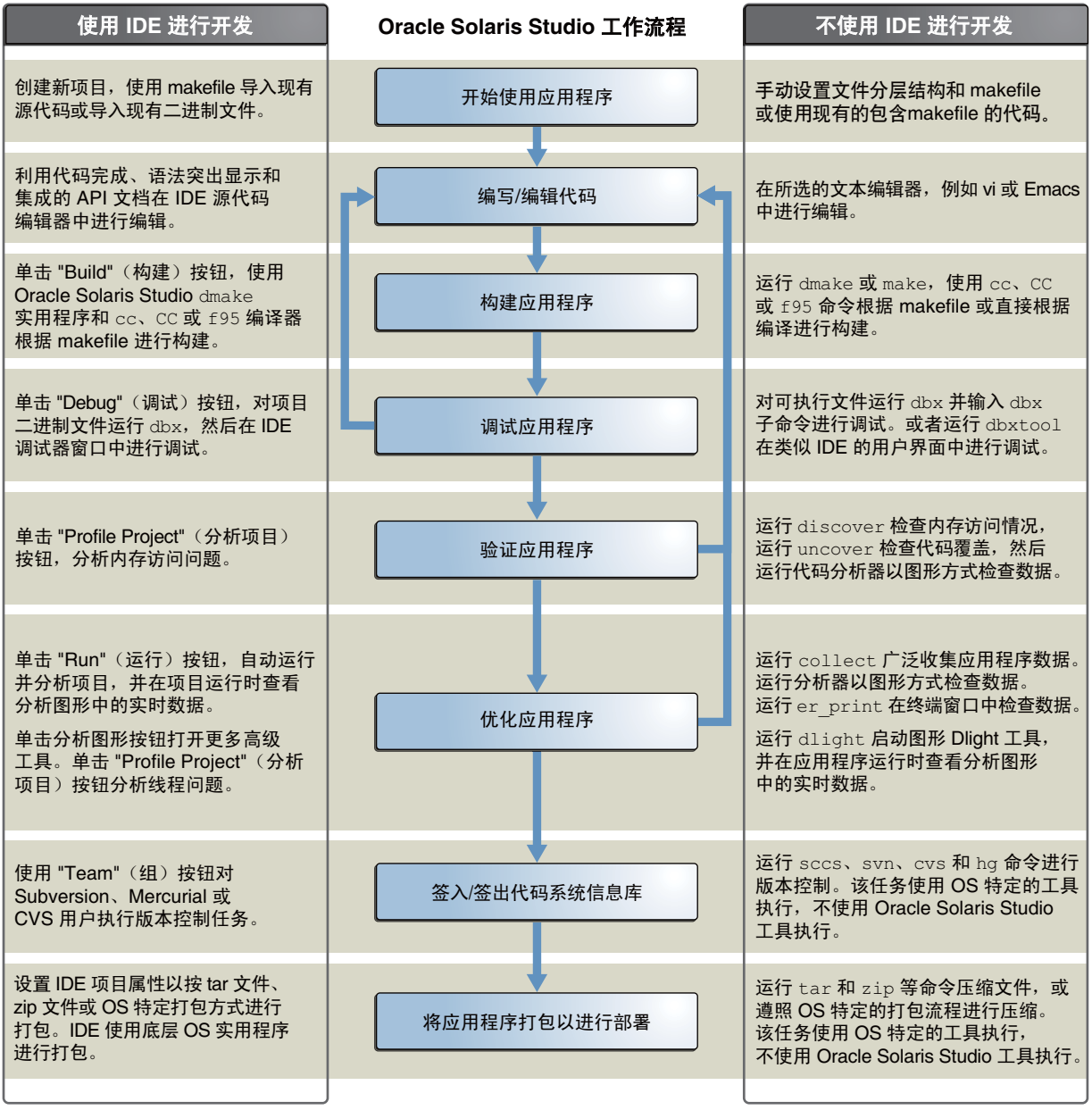
# Oracle Solaris Studio 的开发者工作流程

Oracle Solaris 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。如果愿意，您可以在 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。您可以对在 IDE 中创建的项目的二进制文件使用性能分析器或 DLight。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，可以继续使用您惯用的环境而忽略 IDE，但可以采用 Oracle Solaris Studio 编译器和性能工具，以使您的应用程序在使用 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 Solaris Studio 工具的开发者工作流程。




本文档的其余内容介绍 Oracle Solaris 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 Solaris Studio IDE

Oracle Solaris Studio IDE 以 NetBeans IDE 这种开源集成开发环境为基础。Oracle Solaris 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 Solaris Studio 组件。

Oracle Solaris Studio IDE 使用 Oracle Solaris 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 Solaris DTrace 和 Oracle Solaris Studio 性能实用程序来收集运行中项目的数据。

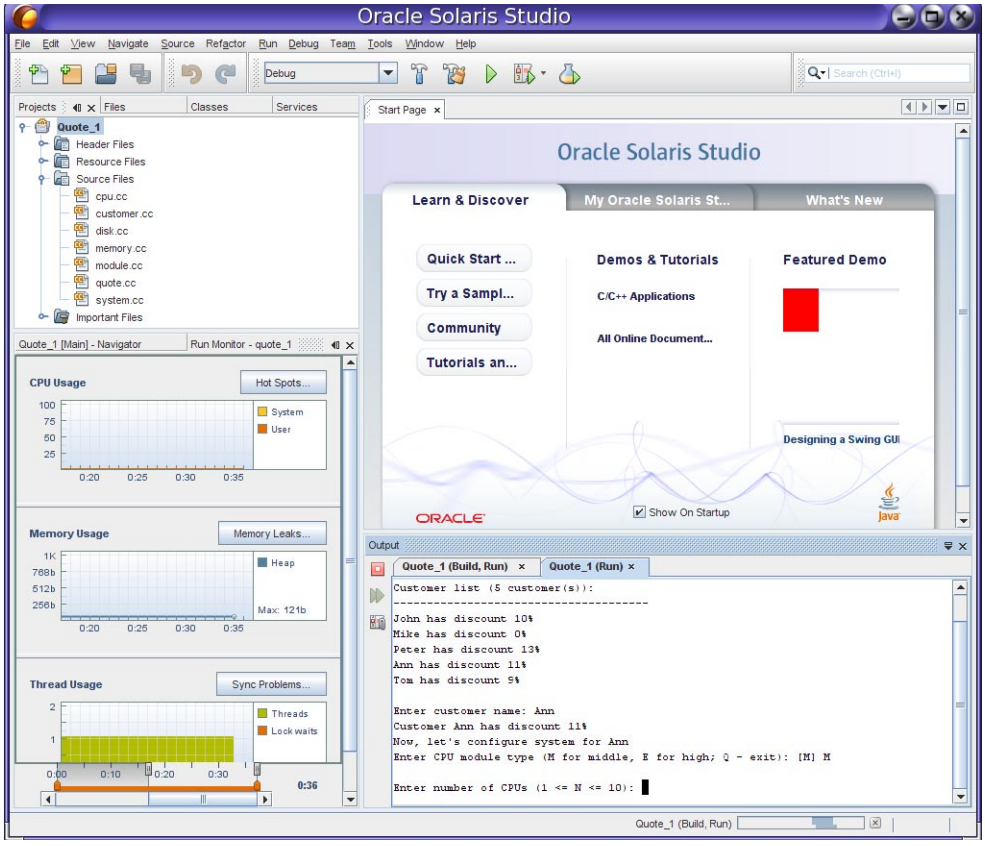
与使用文本编辑器和命令行进行开发相比，使用 Oracle Solaris Studio IDE 具有以下优点：

- **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 IDE 功能非常有用，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**。您可以在桌面系统上运行 IDE，而使用安装在远程服务器上的 Oracle Solaris Studio 编译器和工具。工具在远程服务器上运行，同时显示给在您的本地系统上运行的 IDE，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：

```
% solstudio
```

下图说明了正在运行 Quote 示例项目的 IDE。




C 或 C++ 或 Fortran 项目是一组源文件以及有关如何编译及链接这些源文件并运行生成的程序的相关信息。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，这样 IDE 才能构建项目。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 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 的集成帮助，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。

《Oracle Solaris Studio 12.3 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++ 学习资源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 Solaris Studio 编译器

Oracle Solaris 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 Solaris 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 Solaris 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 Solaris 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 Solaris Studio C 编译器符合 C 编程语言标准 ISO/IEC 9899:1999 和 C 编程语言标准 ISO/IEC 9899:1990 标准。C 编译器还支持 OpenMP 3.1 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 [-llibrary]...
```

编译器选项附加在源文件名称前面。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。

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。

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 Solaris Studio 12.3：C 用户指南》和 cc (1) 手册页。有关新功能和已更改功能、软件更正、问题和解决方法以及编译器的限制和不兼容性的信息，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

# C++ 编译器

Oracle Solaris Studio C++ 编译器 (CC) 支持 C++ ISO 国际标准 ISO IS 14822:2003 编程语言 — C++。CC 编译器还支持 OpenMP 3.1 共享内存并行 API。OpenMP 3.1 API 随 Oracle Solaris Studio 12.3 提供。

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 [-llibrary]...
```

编译器选项附加在源文件名称前面。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 来结尾。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。

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 Solaris Studio 12.3：C++ 用户指南》和 CC(1) 手册页。有关新功能和已更改功能、软件更正、问题和解决方法以及编译器的限制和不兼容性的信息，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

# Fortran 95 编译器

Oracle Solaris 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 3.1。

`f95` 命令调用 Oracle Solaris 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 程序的示例命令：

```
% 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 Solaris Studio 12.3：Fortran 用户指南》和 `f95`(1) 手册页。有关新功能和已更改功能、软件更正、问题和解决方法以及编译器的限制和不兼容性的信息，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

# C/C++/Fortran 库

Oracle Solaris 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 C++ 运行时 libCrun 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 可以查看该手册页。

多数 /usr/lib 库都有 C 接口。缺省情况下，libc 和 libm 库由 cc 和 CC 编译系统链接，因此无需显式链接这两个库。要链接任何其他 /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。

Fortran 运行时库随 Oracle Solaris Studio 而非操作系统提供。

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》。

有关链接库的详细信息，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 3.1

Oracle Solaris Studio 编译器符合 OpenMP 3.1 共享内存并行化 API 规范。OpenMP 由可以用来开发多线程应用程序的一组编译器指令、库例程和环境变量构成。

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。要在多线程环境中运行并行化程序，必须在执行之前设置 OMP_NUM_THREADS 环境变量，以指定程序可以使用的线程数。如果未指定 OMP_NUM_THREADS，则缺省为 2。有关详细信息，请参见《Oracle Solaris Studio 12.3： OpenMP API 用户指南》。

# 适合密集计算程序的 Sun 性能库

Sun 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。Sun 性能库基于公共域子例程集合，其可从 Netlib 获得，网址为 http://www.netlib.org。Sun 增强了这些公共域子例程，并将其捆绑成 Sun 性能库。

Sun 性能库例程在串行平台和多处理器 (MP) 平台上都能够提高应用程序的性能，这是因为很多 Sun 性能库例程的串行速度都已得到提高，而且很多例程都已进行了并行化处理。Sun 性能库例程还具有在基本 Netlib 库中未提供的 SPARC 和 AMD 特定优化，包括支持 Fortran 95 和 C 语言接口的扩展。

Sun 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 Sun 性能库的系统中，必须使用 `-library=sunperf` 和 `-staticlib=sunperf` 选项。

有关使用 Sun 性能库的完整信息，请参见《Oracle Solaris Studio 12.2：Sun 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 Oracle Solaris Studio 手册页的 3p 部分。要了解 Sun 性能库当前版本的新功能和已更改功能、软件更正、已知问题、限制以及不兼容性，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

# 用于生成应用程序的 dmake 实用程序

`dmake` 实用程序是一种命令行工具，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。

缺省情况下，当您使用项目的 makefile 中的目标在 IDE 中生成并运行项目时，Oracle Solaris Studio IDE 将使用 `dmake`。也可以使用 `dmake` 通过 IDE 执行各个 makefile 目标。如果愿意，您可以将 IDE 配置为使用 `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 Solaris Studio 12.3：分布式创建 (dmake)》或 `dmake`(1) 手册页。要获得当前版本 `dmake` 中的新功能和已更改功能、软件更正、已知问题、限制以及不兼容性的列表，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

# 用于调试应用程序的工具

Oracle Solaris 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 Solaris 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 的详细信息，请参见以下部分：


- 第 20 页中的 “命令行中的 dbx”
- 第 21 页中的 “IDE 中的 dbx”
- 第 22 页中的 “dbxtool 中的 dbx”


## 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)
```

当执行过程在断点处停止时，您可以键入 `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。

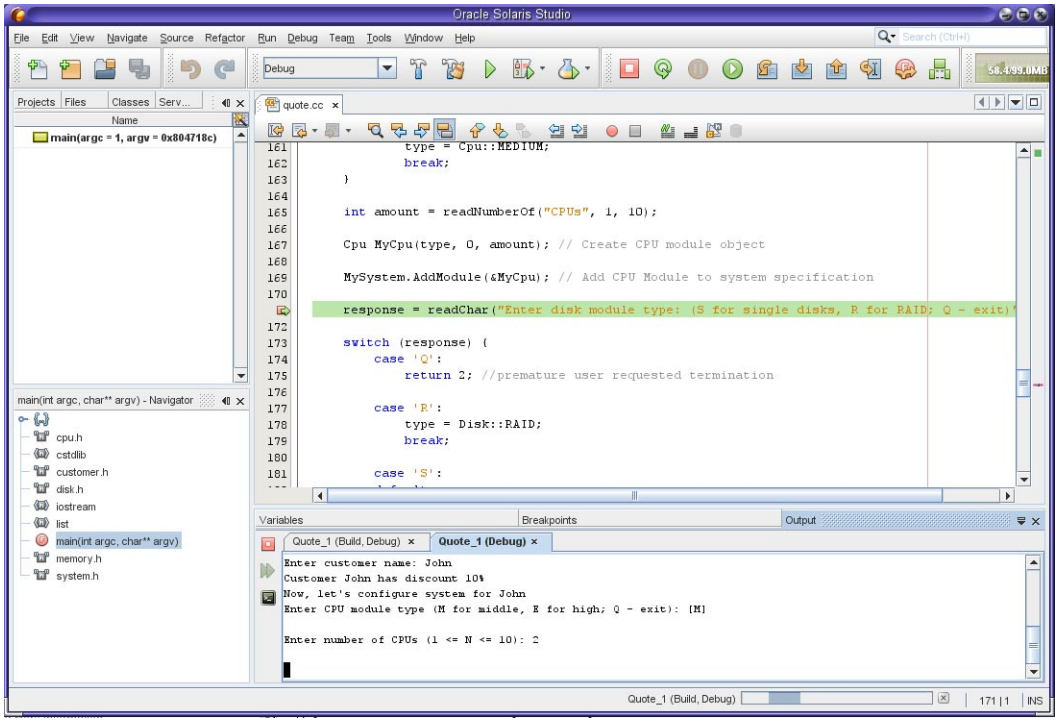
有关 `dbx` 实用程序的命令行选项的信息，请参见 `dbx(1)` 手册页。

有关使用 `dbx` 的完整信息（包括命令参考部分），请参见《Oracle Solaris Studio 12.3：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。要获得当前版本 `dbx` 中的新功能和已更改功能、软件更正、已知问题、限制以及不兼容性的列表，请参见《Oracle Solaris Studio 12.3 发行版的新增功能》。

## IDE 中的 dbx

您可以在 Oracle Solaris 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。在下面的屏幕抓图中，`dbx` 中正运行一个 IDE 示例项目。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。当使用 "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 的详细信息，请参见 IDE 中的集成帮助和《Oracle Solaris Studio 12.3 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用户界面。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。

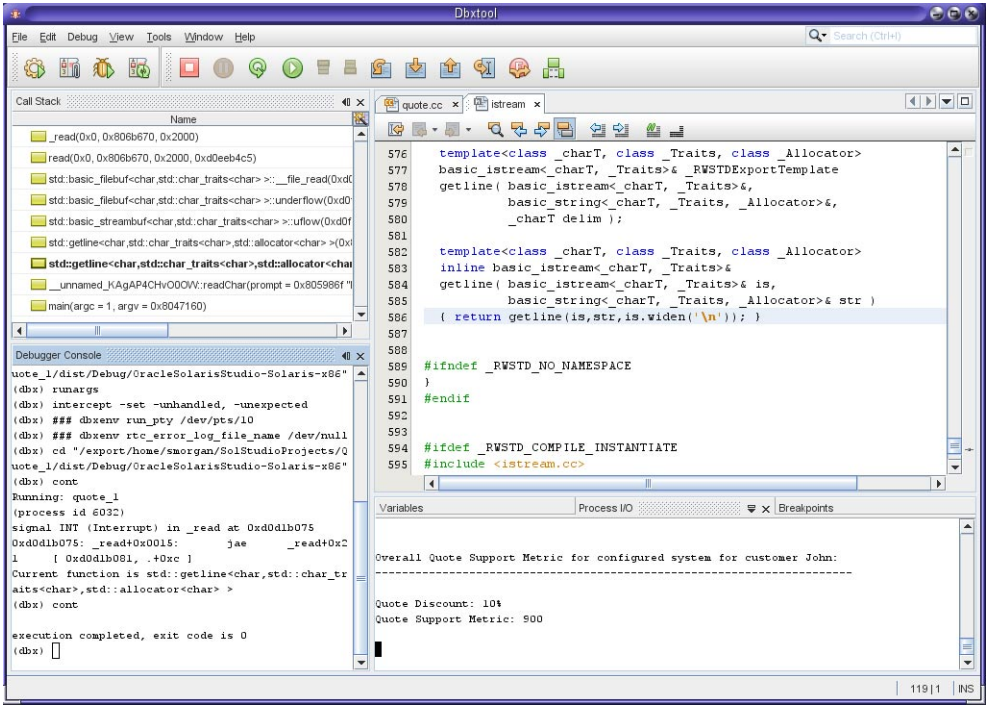
要启动 dbxtool，请键入：

```
% dbxtool executable-name
```

您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 Solaris Studio 12.3： dbxtool 教程》中介绍如何使用 dbxtool。

# 用于验证应用程序的工具

Oracle Solaris 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。

## 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 简单易用。在使用 `-g` 选项编译二进制文件后，您可以对二进制文件运行 `discover` 命令已对其进行检测。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` 选项指出报告应采用文本格式，`-` 选项则指出应输出到屏幕上。

```
% 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 Solaris Studio 12.3： 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 Solaris 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。

编译二进制文件后，对该二进制文件运行 `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 Solaris Studio 12.3： Discover 和 Uncover 用户指南》。

# 用于集成错误检查的代码分析器工具

Oracle Solaris 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：

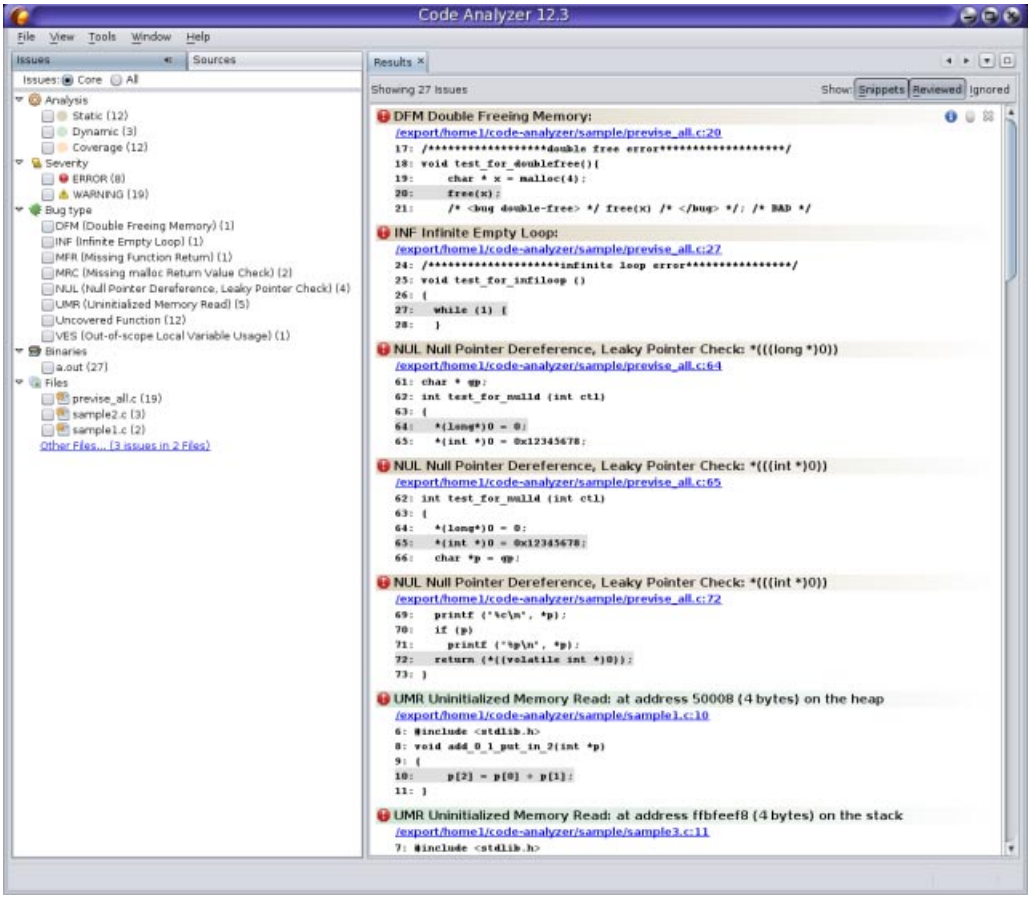
- 静态代码检查，当您使用 Oracle Solaris Studio C 或 C++ 编译器并指定 -xanalyze=code 选项来编译应用程序时，执行该检查。
- 动态内存访问检查，当您使用带有 -a 选项的 Discover 来检测二进制文件，然后运行检测后的二进制文件时，执行该检查。
- 代码覆盖检查，当您使用 Uncover 检测二进制文件，运行检测后的二进制文件，然后对收集的覆盖数据运行带有 -a 选项的 Uncover 时，执行该检查。

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。




# 用于调整应用程序性能的工具

Oracle Solaris Studio 软件提供了多个可用于检查应用程序行为的工具，从而使您能够对性能调优。

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问题所在的位置。
- **简单性能优化工具 (Simple Performance Optimization Tool, SPOT)。**一种命令行工具，它与性能分析器工具配合使用，生成 Web 页来报告这些工具收集的数据。
- **DLight 中的分析工具。**一组同时运行的图形工具，使您能够以同步方式从多个源分析有关正在运行的应用程序的数据。

- **IDE 中的分析工具**。与 DLight 中的分析工具类似，使您能够在 IDE 中检查项目的性能。

# 性能分析器工具

Oracle Solaris 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。

您可以选择使用 Oracle Solaris 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。

相对于 DLight 分析工具，利用性能分析器工具，对所收集数据具有更大的控制能力、检测数据更为深入，同时还能检查程序与硬件之间的交互。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. 使用收集器收集性能数据。
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检查数据，检测多线程应用程序上是否存在数据争用和死锁。

## 使用收集器收集性能数据

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。可通过使用 `collect` 命令、从性能分析器中或使用 `dbx` 调试器的 `collect` 子命令来运行收集器。

Oracle Solaris 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 `-c on` 选项时，收集器将记录指令计数。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 ...
00:00:00.000  =====  (15909) synprog run
00:00:00.002  =====  (15909) Thu  10 Nov 11  15:12:18 Stopwatch calibration
  OS release 5.10 -- enabling microstate accounting 5.10.
        0.001498 s.  (32.8 % of 0.004568 s.) -- inner
        N = 1000, avg = 1.498 us., min = 0.721, max = 596.665
        0.003482 s.  (72.9 % of 0.004776 s.) -- outer
        N = 1000, avg = 3.482 us., min = 2.883, max = 599.007
00:00:00.007  =====  (15909)  Begin commandline
        icpu.md.cpu.rec.recd.dousl.gpf.fitos.uf.ec.tco.b.nap.sig.sys.so.sx.so
00:00:00.008  =====  (15909) start of icputime
    3.019055 wall-secs.,   2.328491 CPU-secs., in icputime
00:00:03.027  =====  (15909) start of muldiv
    3.012635 wall-secs.,   2.675769 CPU-secs., in muldiv
00:00:06.040  =====  (15909) start of cputime
    3.000567 wall-secs.,   2.591964 CPU-secs., in cputime
00:00:09.041  =====  (15909) start of recurse
    ...
    (output edited to conserve space)
    ...
```

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。

有关如何使用收集器的信息，请参见性能分析器中的 "Help"（帮助）菜单、《Oracle Solaris Studio 12.3：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

性能分析器是一个图形用户界面 (graphical user interface, GUI)，显示收集器记录的数据的度量。这些度量包括：

- 时钟分析度量，按多种类别说明程序消耗时间的位置。
- 硬件计数器度量，显示程序所遇到的 CPU 特定事件的相关信息。
- 同步延迟度量，显示由多线程程序的不同线程执行的任务同步中的延迟。
- 内存分配度量，显示程序中的内存泄漏。
- MPI 跟踪度量，帮助您确定 MPI 程序中可能因 MPI 调用而出现性能问题的位置。

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：

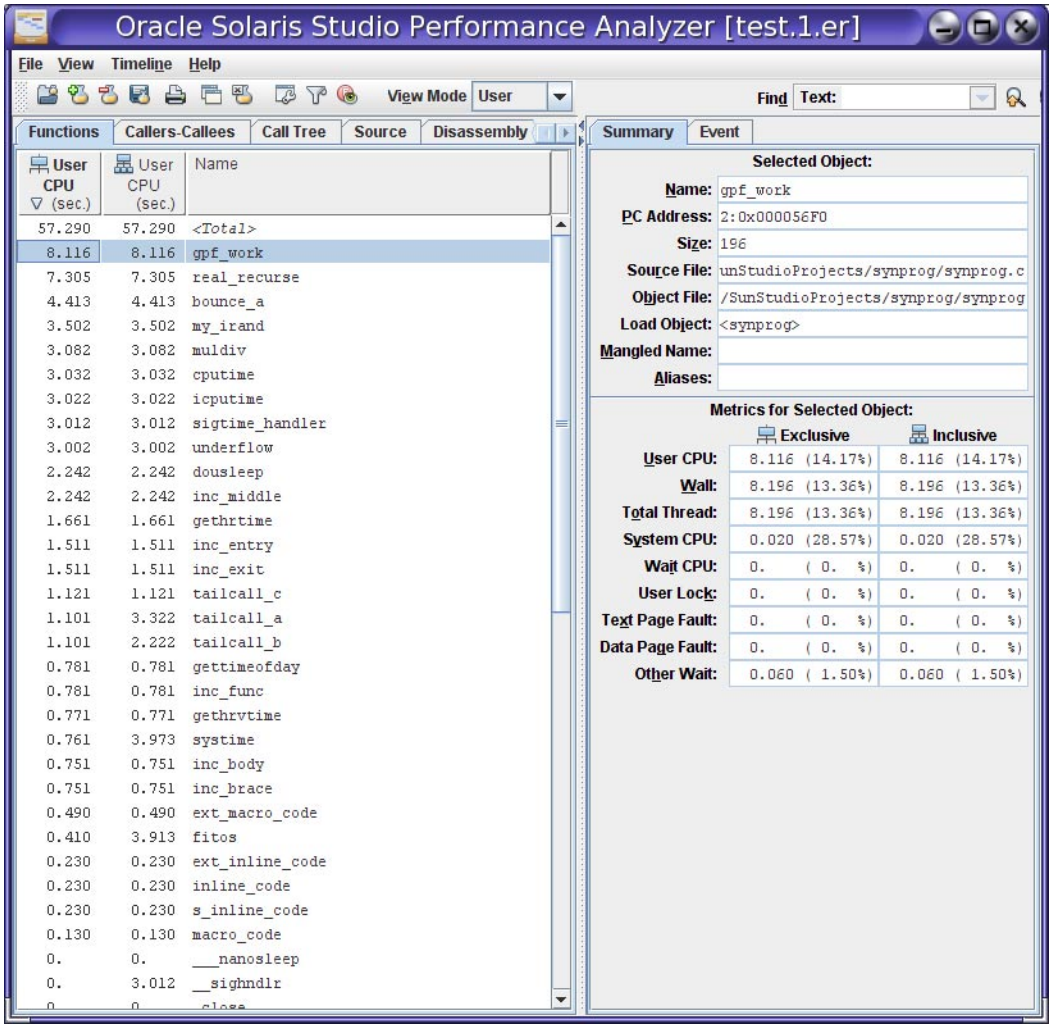
```
% analyzer [experiment-list]
```

*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。对多个实验进行调用时，缺省情况下，分析器将聚集实验数据，但也可用于对实验进行比较。

要在性能分析器中打开实验 `test.1.er`：

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

下图显示性能分析器 "Functions"（函数）标签，对 `synprog` 示例进行 `test.1.er` 实验。"Functions"（函数）标签显示由 `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 "Summary"（摘要）标签显示 `gpf_work` 函数资源使用情况的详细信息。




有关使用性能分析器的信息，请参见《Oracle Solaris Studio 12.3：性能分析器》手册、性能分析器集成帮助以及 `analyzer(1)` 手册页。

## 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 User CPU Time

Excl.     Incl.      Name
User CPU  User CPU
  sec.      sec.
57.290    57.290     <Total>
 8.116     8.116     gpf_work
 7.305     7.305     real_recurse
 4.413     4.413     bounce_a
 3.502     3.502     my_irand
 3.082     3.082     muldiv
 3.032     3.032     cputime
 3.022     3.022     icputime
 3.012     3.012     sigtime_handler
 3.002     3.002     underflow
 2.242     2.242     dousleep
 2.242     2.242     inc_middle
 1.661     1.661     gethrtime
 1.511     1.511     inc_entry
 1.511     1.511     inc_exit
 1.121     1.121     tailcall_c
 1.101     3.322     tailcall_a
 1.101     2.222     tailcall_b
 0.781     0.781     gettimeofday
 0.781     0.781     inc_func
 0.771     0.771     gethrvtime
 0.761     3.973     systime
 0.751     0.751     inc_body
 0.751     0.751     inc_brace
 0.490     0.490     ext_macro_code
.
.       lines deleted
```

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 Solaris Studio 12.3：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、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。

线程分析器针对多线程程序分析经过简化，仅显示性能分析器的 "Races"（争用）、"Deadlocks"（死锁）、"Dual Source"（双源）、"Race Details"（争用详细信息）和 "Deadlock Details"（死锁详细信息）标签。对于 OpenMP 程序，还显示 "OpenMP Parallel Region"（OpenMP 并行区域）和 "OpenMP Task"（OpenMP 任务）标签。

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。

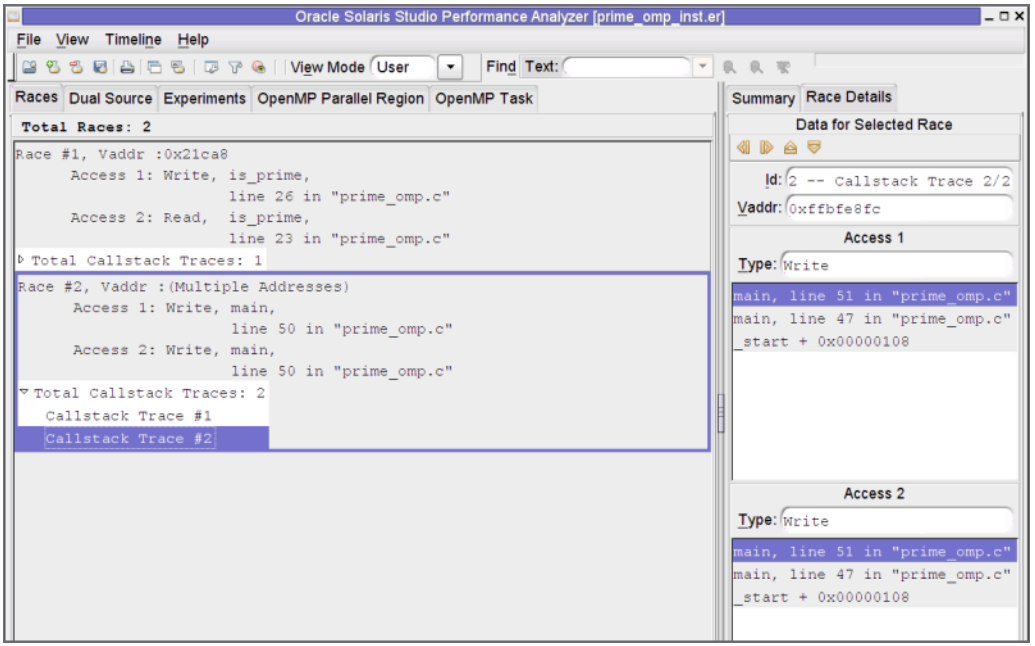
要使用线程分析器：

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 Solaris Studio 12.3：线程分析器用户指南》。

# 简单性能优化工具 (Simple Performance Optimization Tool, SPOT)

简单性能优化工具 (Simple Performance Optimization Tool, SPOT) 可帮助您诊断应用程序中的性能问题。SPOT 对应用程序运行一组性能工具并生成 Web 页来报告这些工具收集的数据。也可独立于 SPOT 运行这些工具。

SPOT 是对 Oracle Solaris Studio 性能分析器的补充。性能分析器会告诉您应用程序运行过程中耗用时间的位置。不过，在某些情况下，您可能需要更多信息来帮助诊断应用程序的问题。SPOT 便能在此时为您提供帮助。

SPOT 使用性能分析器的 `collect` 实用程序作为它的工具之一。SPOT 使用 `er_print` 实用程序和另外一个实用程序 `er_html` 以 Web 页的形式显示分析数据。

使用 SPOT 之前，应通过某种程度的优化（使用 `-O` 选项）和调试信息（使用 `-g` 选项）对应用程序二进制文件进行编译，以使 SPOT 工具将性能信息体现到代码行中。

通过启动应用程序或连接到正在运行的应用程序，可使用 SPOT 收集性能数据。

要运行 SPOT 并启动您的应用程序：

```
% spot executable
```

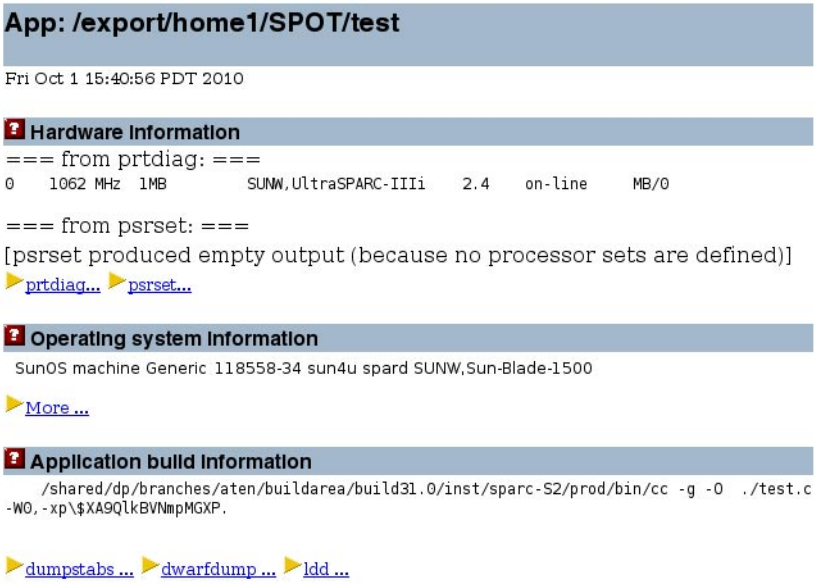
要对已经运行的应用程序运行 SPOT：

```
% spot -P process-id
```

SPOT 在每次运行应用程序时都会生成一个报告，还会生成一个将各次运行产生的 SPOT 数据进行比较的报告。

如果对 PID 使用 SPOT，会按顺序将多个工具连接到 PID 以生成报告。

下图显示了 SPOT 运行报告的一部分，其中显示有关运行 SPOT 的系统及应用程序编译情况的信息。该报告包括指向含有更多信息的其他页面的链接。




SPOT 报告 Web 页面相互链接在一起，便于您检查编译的所有数据。

有关更多信息，请参见《《Oracle Solaris Studio 12.2: Simple Performance Optimization Tool (SPOT) User's Guide》》。

# DLight 中的分析工具

DLight 是一种交互式图形工具，采用 Oracle Solaris 动态跟踪 (DTrace) 技术监测正在运行程序的行为。DLight 同时启动多个分析工具，使您能够以同步方式从多个源分析有关正在运行的应用程序的数据。DLight 分析工具能帮助您确定应用程序中出现运行时问题的根本原因。这些工具的影响低，执行分析时不会对程序或系统造成负面影响。

DLight 分析工具需要用于控制用户对 DTrace 功能的访问的特权。因此，应该在您具有管理特权或具有管理员授予您的 `dtrace_user`、`dtrace_proc` 和 `dtrace_kernel` 特权的系统上运行 DLight。

要启动 DLight：

```
% dlight
```

选择想要监视的目标应用程序以及希望 DLight 运行的分析工具。目标应用程序可在本地系统上运行，也可在您具有登录访问权限和 DTrace 特权的远程联网系统上运行。

可对尚未运行的可执行文件运行 DLight，也可将其连接到正在运行的进程。还可以将 DLight 连接到包括一个进程以及该进程启动的所有子进程的进程树。当目标程序运行时，DLight 以图形方式显示它所收集到的数据。

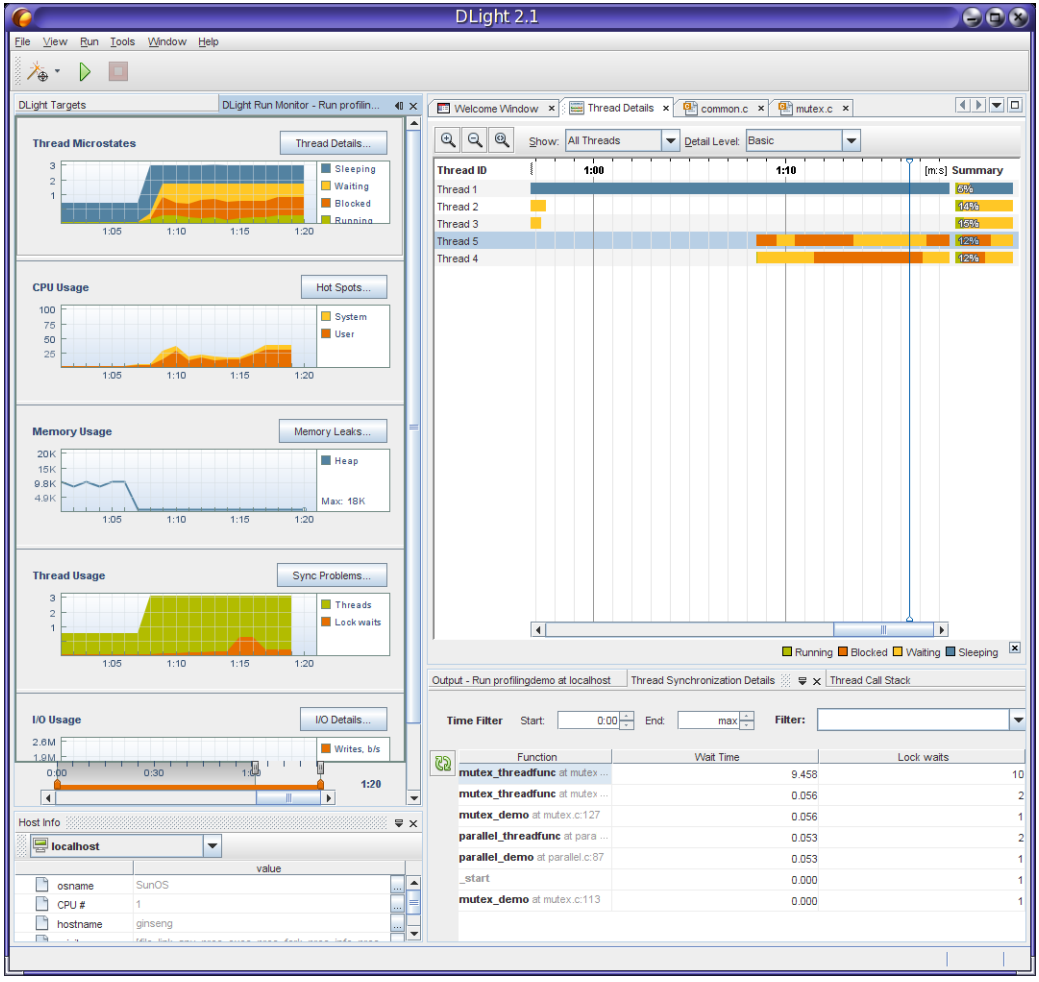
DLight 包括以下分析工具：

- 线程微状态－显示当程序运行时其中运行的线程状态的相关摘要数据。
- CPU 使用情况－显示程序运行期间该程序所使用的 CPU 时间的百分比。CPU 时间分为用户 CPU 时间和系统 CPU 时间。
- 内存使用情况－显示程序的内存堆如何随时间变化。
- 线程使用情况－显示程序中运行的线程数并指明线程在何时处于等待状态。
- I/O 使用情况－显示程序读写的字节数。

每个工具都提供了一个按钮，用于打开显示有更多详细信息的相关工具：

- "Thread Details"（线程详细信息）－在 "Thread Microstates"（线程微状态）图形中单击 "Thread Details"（线程详细信息）按钮
- "CPU Time Per Function"（每个函数的 CPU 时间）－在 "CPU Usage"（CPU 使用情况）图形中单击 "Hot Spots"（热点）按钮
- "Memory Leaks"（内存泄露）－在 "Memory Usage"（内存使用情况）图形中单击 "Memory Leaks"（内存泄露）按钮
- "Thread Synchronization Details"（线程同步详细信息）－在 "Thread Usage"（线程使用情况）图形中单击 "Sync Problems"（同步问题）按钮
- "I/O Details"（I/O 详细信息）－在 "I/O Usage"（I/O 使用情况）图形中单击 "I/O Details"（I/O 详细信息）按钮

下图显示针对《Oracle Solaris Studio 12.3： DLight 教程》中使用的 ProfilingDemo 示例应用程序运行的 DLight 分析工具。

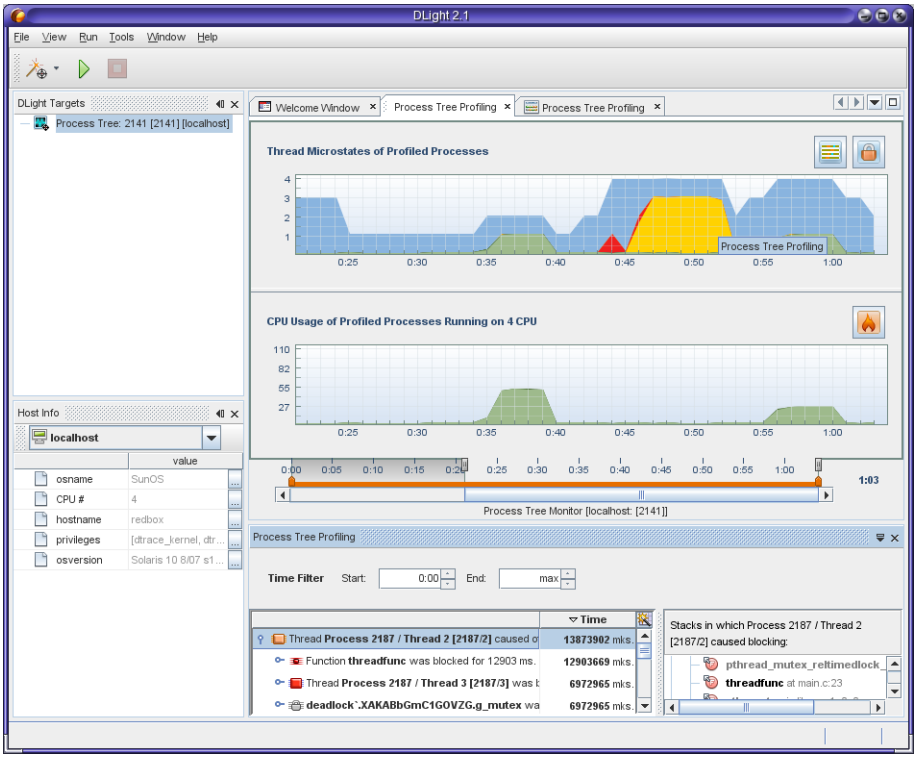



在图中，当用户单击 "Thread Microstates"（线程微状态）工具中的 "Thread Details"（线程详细信息）按钮后，"Thread Details"（线程详细信息）窗口在 DLight 窗口的左上方打开。当用户单击 "Thread Usage"（线程使用情况）工具中的 "Sync Problems"（同步问题）按钮后，将打开位于右下方的 "Thread Synchronization Details"（线程同步详细信息）窗口。

对进程树目标运行 DLight 时，将显示以下工具：

- 已分析进程的线程微状态－显示 DLight 使用 "Process Tree Target"（进程树目标）分析的所有进程的所有线程的微状态集合。此图形中的按钮将打开关于以下详细信息的窗口：
- 进程树微观状态详细信息－显示目标进程及其子进程的各线程的时间线形式的微状态转换。在 "Thread Microstates of Profiled Processes"（已分析进程的线程微观状态）工具中单击按钮可打开此工具。
- 进程树已阻塞线程详细信息－显示进程及其子进程的锁定统计信息。在 "Thread Microstates of Profiled Processes"（已分析进程的线程微观状态）工具中单击按钮可打开此工具。
- 已分析进程的 CPU 使用情况－显示在运行目标进程的所有 CPU 分析的所有目标进程的所有线程的累积 CPU 使用情况。
- 进程树 CPU 热点－通过显示程序中的函数以及该函数及其调用的所有函数使用的 CPU 时间来显示程序进程树中大量使用 CPU 的区域。在 "CPU Usage of Profiled Processes Running"（运行的已分析进程的 CPU 使用情况）工具中单击按钮可打开此工具。

下图显示针对进程树目标运行的 DLight。




图中底部显示进程树已阻塞线程详细信息窗口，该窗口由用户单击 "Thread Microstates of Profiled Processes"（已分析进程的线程微观状态）工具中的锁图标打开。在此窗口的右侧，可以看到发生所选线程锁定的调用堆栈。

这些 "Process Tree Profiling"（进程树分析）图形共同用于确定应用程序的多个进程和多个线程协同工作的方式。您能够看到线程被阻塞的位置以及对 CPU 使用情况的影响，并能够将问题的范围缩小到发生这些问题的代码行。

有关如何使用 DLight 的信息，请参见《Oracle Solaris Studio 12.3： DLight 教程》和 "Help"（帮助）菜单中提供的 DLight 的集成帮助。

# IDE 中的分析工具

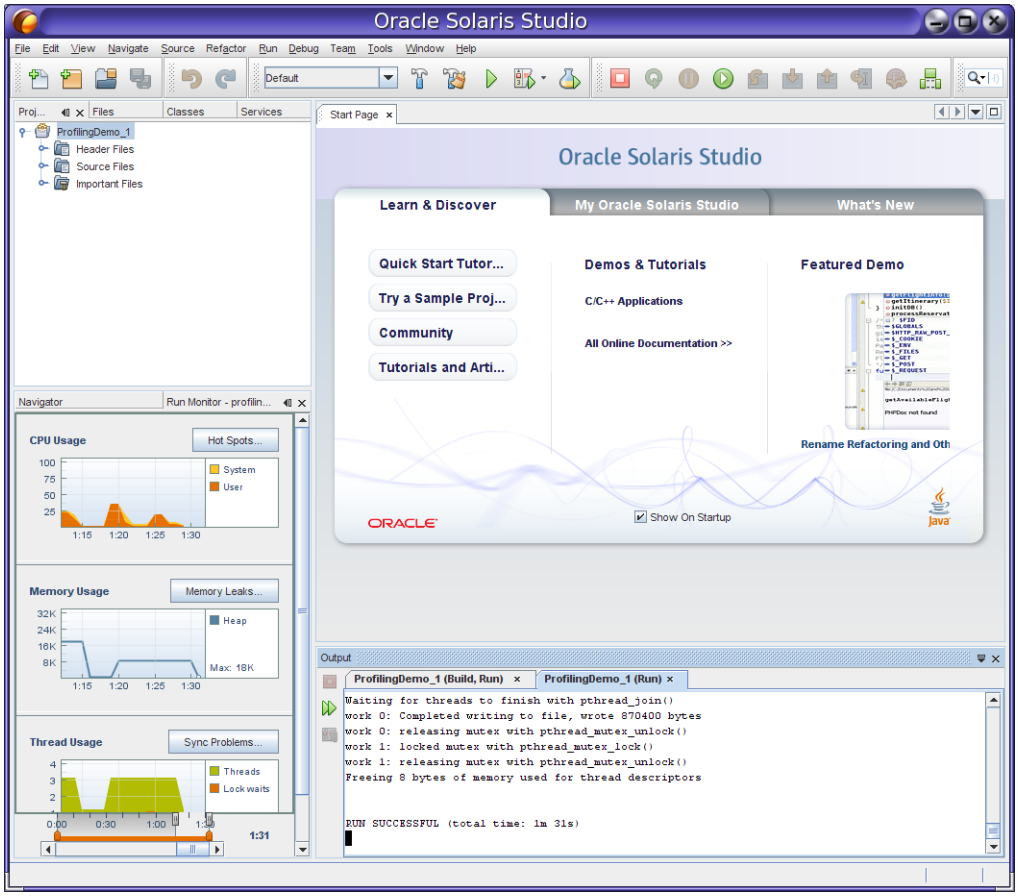
IDE 提供的许多分析工具都与 DLight 相同，使您能够在 IDE 中检查项目的性能。只要您运行 C、C++ 或 Fortran 项目，这些工具就会自动运行。这些工具的影响低，执行分析时不会对程序或系统造成负面影响。

数据以图形方式呈现，方便您查看程序对资源使用情况的摘要。运行项目时，"Run Monitor"（运行监视器）窗口会自动打开，显示这些影响较低的工具的输出。可根据需要禁用分析工具，或指定您希望自动运行的那些工具。

缺省的分析工具不将 DTrace 用作底层技术。而是使用 Studio 实用程序和操作系统实用程序收集数据。这种方法使得所有用户都能够使用这些工具，无论他们运行的是 Oracle Solaris 还是 Linux。不过，如果在 Oracle Solaris 上运行 IDE，也可以选择使用 DTrace 的工具并提供更详细的信息。

与在 DLight 中相同，使用 DTrace 的 IDE 工具需要用于控制用户对 DTrace 功能的访问的特权。有关如何分配特权的信息，请参见 IDE 帮助中的说明 "Enabling DTrace for Profiling C/C++/Fortran Applications"（为分析 C/C++/Fortran 应用程序启用 DTrace）。

下图显示使用缺省运行监视器工具的 IDE。

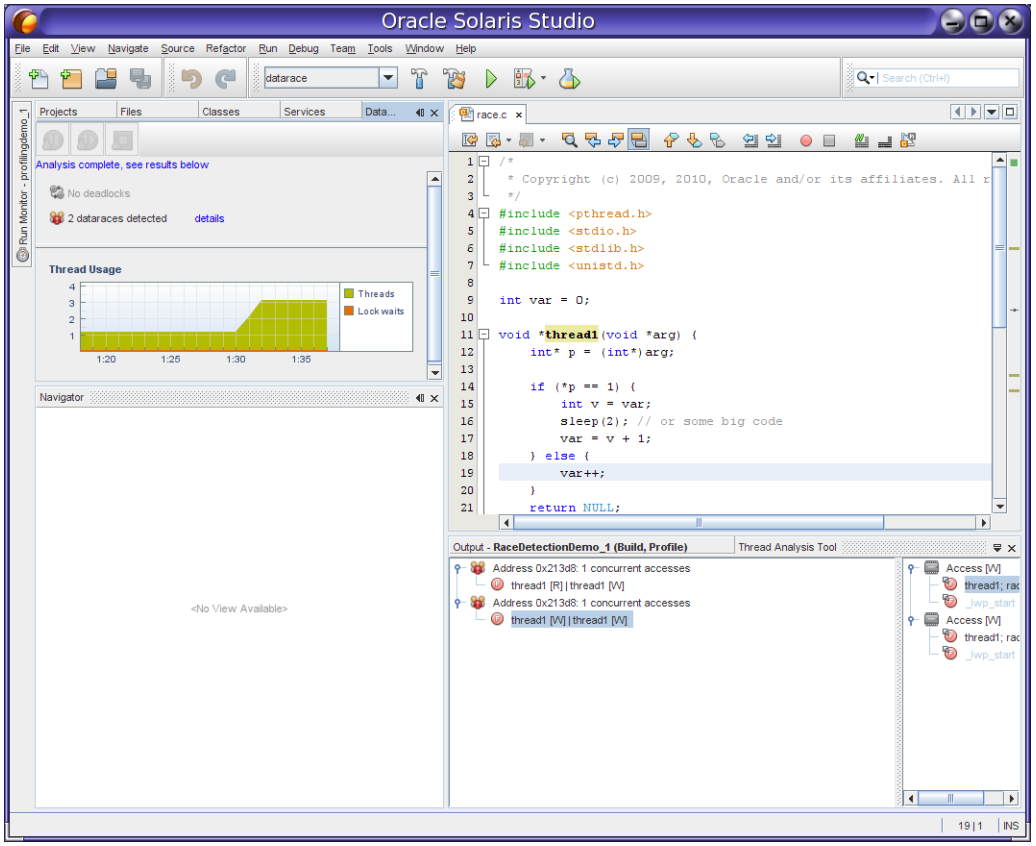



用于执行更详细分析的其他工具对系统和应用程序性能的影响较大，因此不会自动运行这些工具。高级工具与自动分析工具相链接，只需单击按钮便可轻松启动。

IDE 具有 DLight 中没有的另外两个工具："Data Races and Deadlocks Detection"（数据争用和死锁检测）工具和 "Memory Access Error"（内存访问错误）工具。

"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）。

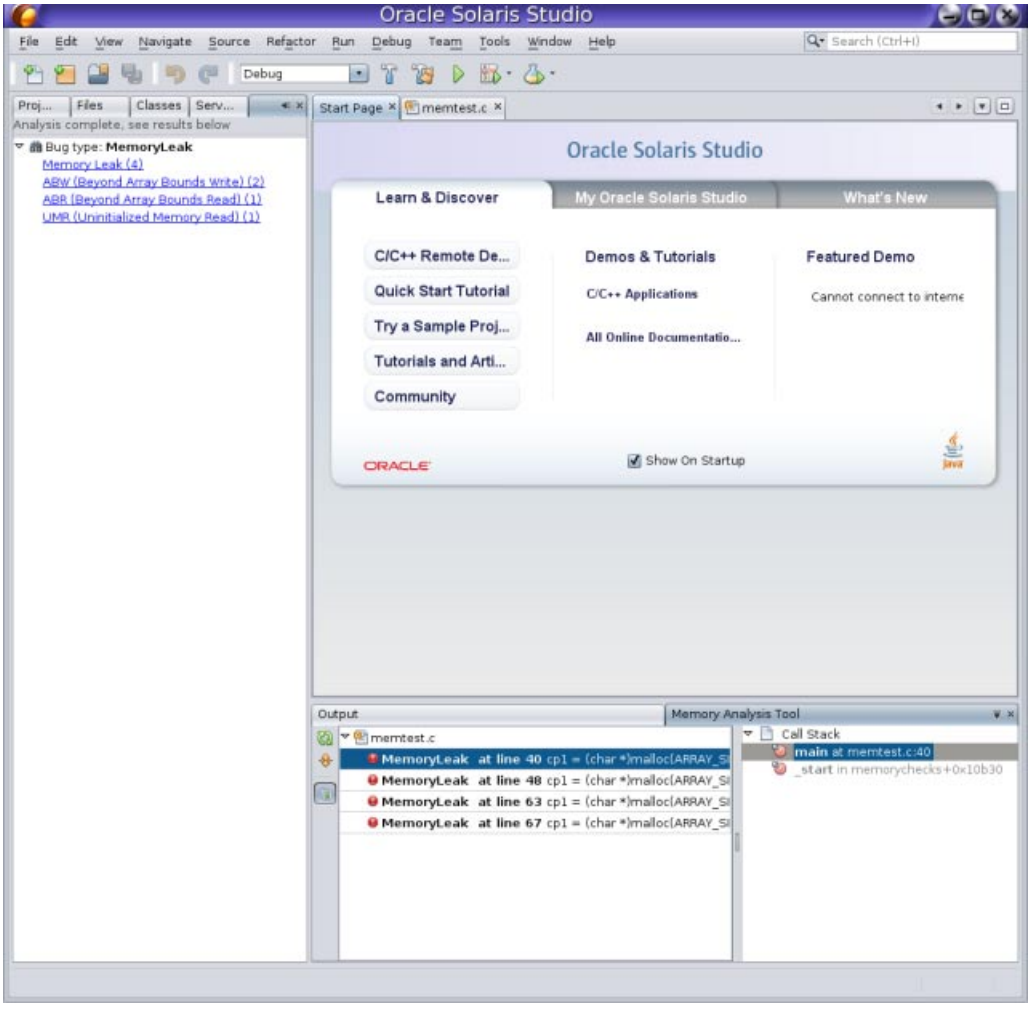
下图显示在检测到数据争用之后的 "Data Races and Deadlocks Detection"（数据争用和死锁检测）工具。




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）。

# 更多信息

有关更多信息，请参见 Oracle 技术网上的 Oracle Solaris 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，以从 Oracle Solaris Studio 受益更多。