Oracle® Developer Studio 12.6 发行版的新增功能

文件号码 **E83252-01**
2017 年 7 月

ORACLE

文件号码 E83252-01

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

- **概述**－介绍了此 Oracle Developer Studio 12.6 发行版中提供的编译器和工具的新增和更改的功能。
- **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**－编程经验、软件开发测试以及构建和编译软件产品的能力

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。

♦ ♦ ♦ 第 1 章

Oracle Developer Studio 12.6 发行版简介

本章概述了此发行版中的主要更新。

Oracle Developer Studio 概述

Oracle Developer Studio 包括编译器套件、分析套件以及图形集成开发环境 (integrated development environment, IDE)，该环境专为这两个套件中的编译器和工具而设计。它们一起提供了一个开发环境，该环境已进行了优化，可以用来开发在 Oracle Sun 硬件上具有最佳性能的应用程序。

图 1 图中显示了与 IDE 集成的编译器和分析套件

此发行版中的主要功能

Oracle Developer Studio 12.6 提供高度优化的编译器、高级分析工具和可以识别多种语言的 IDE，以便轻松开发适用于本地或云中的 Oracle Solaris 和 Linux 操作系统的快速、可靠且安全的应用程序。Oracle Developer Studio 工具经过优化，可为整个硬件和软件堆栈提供有力补充，并支持开发团队更快速地编写更优质的代码。

下面概述了此发行版中的主要功能：

- **生成在 Oracle 系统和 Oracle Cloud 中运行速度最快的代码** — 编译器、代码生成器和运行时库进行了调优，以利用最新的 Oracle SPARC M8、Fujitsu M12 和基于 Intel

Xx86 Skylake 的服务器。这些新系统或处理器引入了新指令、新的高速缓存大小以及新管道。对于在这些新系统上运行的应用程序，编译器在编译时会考虑这些因素，以生成运行速度最快的代码，从而使这些应用程序实现最佳的性能。有关详细信息，请参阅编译器用户指南。

- 进行了 SPARC M8/T8 高速缓存和指令调度优化
- 提供了新的 pragma 来支持在 SPARC 上访问未对齐的数据
- 可通过性能分析器访问 SPARC M8 和 Intel Skylake 硬件计数器信息
- 主要性能库例程进行了调优，以利用 SPARC M8 芯片功能
- 进行了 Fujitsu SPARC M12 优化
- 提供了 Intel Skylake AVX512 支持

- **良好的开源兼容性和互操作性使开发过程大大简化**－编译器对正式语言标准的支持至关重要，这样开发者才能在大多数现代数据企业数据中心的各种系统之间迁移代码。此外，开源迁移计划已经催生了大量将要结合到企业系统中的应用程序。此开源代码中的大部分是使用 GNU 工具集编写的，因此与该标准的兼容性也很重要。Oracle Developer Studio 12.6 提供以下支持：
  - 符合 C++ 2011 和 C++ 2014 语言标准。有关更多信息，请参见 第 2 章 C++ 编译器。
  - 超过 40 项 gcc 兼容功能，扩充了之前发行版中提供的 gcc 兼容功能。有关详细信息，请参见“编译器通用的新增和更改的功能” [49]
  - IDE 和性能分析器中的 Java 9 支持
- **借助 Oracle Developer Studio 云服务插件加速云计算** — 随着企业从本地数据中心迁移到集中式的云提供商，将应用程序部署到云变得越来越普遍。开发者也在将其开发环境迁移到集中式的云端数据中心。此发行版提供了多种功能，帮助开发者更轻松地进行此转型。
  - Oracle Developer Studio IDE 提供了一个新插件，用于在本地桌面上运行 IDE并访问 Oracle Developer Studio 云服务。该服务提供问题跟踪、源代码管理、构建服务器以及将应用程序部署到 Oracle Cloud 的简便方法。有关详细信息，请参见第 7 章 Oracle Developer Studio IDE。
  - Oracle 云服务使用 ssh 来访问其基础结构即服务的计算 VM。Oracle Developer Studio 远程性能分析器支持 ssh 隧道，这样开发者可以启动和分析 Oracle 云计算 VM 上运行的性能实验结果。有关远程使用性能分析器的详细信息，请参见《Oracle Developer Studio 12.6: Performance Analyzer》 中的 “Using Performance Analyzer Remotely”
  - 随着越来越多的开发者采用微服务模式，他们开始使用容器技术来托管这些微服务。通过 Oracle Developer Studio IDE 可以创建 Docker 容器，以便将来部署到 Oracle Cloud 或支持此类容器的其他系统上。有关详细信息，请参阅“Docker” [40]。
- **提高应用程序安全性** — 应用程序安全性是现代应用程序必须满足的一项重要要求。Oracle Developer Studio 通过提供以下功能，让开发者可以更轻松地创建更安全的应用程序：

- IDE 中的安全编码助手会突出显示不符合 Oracle Solaris 安全编码建议的代码，并提出可能更安全的替代方案。有关详细信息，请参阅“新增和已更改的 IDE 功能” [27]。
- `discover` 工具可以利用 SPARC 芯片安全内存 (Silicon Secured Memory, SSM) 提供关于错误的应用程序内存访问和内存损坏的实时信息，这些问题是会导致安全违规的典型漏洞。有关更多信息，请参见《Oracle Developer Studio 12.6: Discover and Uncover User’s Guide》 中的 “Hardware-Assisted Checking Using Silicon Secured Memory (SSM)”。
- 编译器可以在每次编译时执行自动代码检查。此功能存在副作用，即在使用 Oracle Developer Studio 12.6 编译器生成代码时可能会看到额外的警告。可使用编译器选项禁用这些检查。

♦ ♦ ♦ 第 2 章

2 C++ 编译器

本章介绍了此 Oracle Developer Studio C++ 编译器发行版中新增和更新的功能。

关于 C++ 编译器

本节提供了 Oracle Developer Studio 12.6 C++ 5.15 编译器发行版中引入的新功能和已更改功能的摘要列表。

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 和 C++ 兼容功能。

C++ 编译器更改

本部分包括“编译器通用的新增和更改的功能” [49]中介绍的更改。

此发行版中特定于 C++ 编译器的新功能如下：

- **新的 `section` 属性**－C++ 编译器现在提供了一个新的 `section` 属性，该属性指定特定节中存在的一个变量或函数。
- 针对 `-xO3` 的**内联行为更改**－支持其正文小于调用方开销的函数的自动内联。此外，当与 `-xspace` 选项一起使用时，`-xO3` 和 `-xO4` 会导致最小的代码大小。
- `-xlibmopt` **选项的更改**－`-xlibmopt` 选项已通过 `-%none`、`-archive` 和 `-shared` 子选项进行了增强。

- **在 Linux 上静态链接的** `libCrunG3`－`libCrunG3` 是通过选项 `-compat=g`、`-std=c++03`、`-std=c++11` 或 `-std=c++14` 中的任意一个编译的 C++ 程序所需的运行时支持库。

  在 Oracle Solaris 上，缺省情况下，`libCrunG3` 在需要时进行动态链接。它是 Oracle Solaris 操作系统的一部分，因此，用户程序始终可以找到它。在 Linux 上，如果您动态链接了 `libCrunG3.so.1`，则需要向该库提供您的程序。为避免此问题，在 Linux 上，`libCrunG3` 缺省情况下将在需要时静态链接。
- **函数重载解决方法中改进的错误报告**－Oracle Developer Studio 和 Oracle Solaris Studio 中以前的编译器在报告对重载的函数的不明确或无法解析的调用时没有提供详细消息。在 Oracle Developer Studio 12.6 中，错误报告已改进。以下示例说明了改进的错误报告功能。

```
Example 1:

% cat ex1.cc
void f(int n, int* ptr);
void f(int, int, int);
void f(int);
void foo()
{
    f(1, 2);
}
```

以前的编译器会报告类似以下内容的消息：

```
% CC -c ex1.cc
"ex1.cc", line 7: Error: Could not find a match for f(int, int) needed in
foo().
1 Error(s) detected.
```

Oracle Developer Studio 12.6 C++ 编译器会提供有关为何无法找到匹配项的更多信息。

```
% CC -c ex1.cc
"ex1.cc", line 7: Error: Could not find a match for f(int, int) needed in
foo().
"ex1.cc", line 1: Note: Candidate 'f(int, int*)' is not viable: argument
'ptr' can't be converted from 'int' to 'int*'.
"ex1.cc", line 3: Note: Candidate 'f(int)' is not viable: too many arguments.
"ex1.cc", line 2: Note: Candidate 'f(int, int, int)' is not viable: too few
arguments.


Example 2:
```

```
% cat ex2.cc
struct A;
struct B {
    B(const A&);
};
struct D1 : B {};
struct D2 : B {};
struct A {
    A();
    operator  D1() const;
    operator  D2&() volatile ;
};
void foo(A& ra)
{
    A va;
    B vb  = ra;
}
```

以前的编译器会报告类似以下内容的消息：

```
% CC -c ex2.cc
"ex2.cc", line 19: Error: Overloading ambiguity between "A::operator D1()
const" and "B::B(const A&)".
"ex2.cc", line 19: Error: Cannot use A to initialize B.
2 Error(s) detected.
```

Oracle Developer Studio 12.6 C++ 编译器会提供有关二义性的更多信息。

```
% CC -c ex2.cc
"ex2.cc", line 19: Error: Type conversion 'A' -> 'B' is ambiguous.
"ex2.cc", line 12: Note: Viable candidate 'A::operator D1() const'.
"ex2.cc", line 4: Note: Viable candidate 'B::B(const A&)'.
"ex2.cc", line 13: Note: Viable candidate 'A::operator D2&() volatile'.
"ex2.cc", line 19: Error: Cannot use A to initialize B.

You can disable generation of the Notes by using the CC option
    -features=no%note
```

有关更多信息，请参见《Oracle Developer Studio 12.6: C++ User's Guide》和 CC(1) 手册页。

♦ ♦ ♦ 第 3 章

3 C 编译器

本章介绍对 Oracle Developer Studio 12.6 发行版的 C 编译器进行的更改。

关于 C 编译器

本节提供了 Oracle Developer Studio 12.6 C 5.14 编译器发行版中引入的新功能和已更改功能的摘要列表。

C 编译器 (cc) 是 C 编译系统的接口。编译过程包括一个预处理器、编译器、代码生成器、优化器、汇编程序和链接编辑器。C 编译器会处理提供的选项，然后通过正确的参数执行各种组件。有关更多信息，请参见 cc(1) 手册页。

C 编译器更改

C 编译器的更改包括“编译器通用的新增和更改的功能” [49]中介绍的更改，以及以下更改：

- 新的编译器选项如下：
  - `-fcommon` 和 `-fno-common`
  - `-features=[no%]gcc_enums`
  - `-fexceptions`
  - `-fsemantic-interposition`
  - `-fno-semantic-interposition`
  - `-fshort-enums`
  - `-fvisibility`
  - `-shared`

  - `-std=gnu11 -std=gnu99 -std=gnu90 -std=gnu89 -std=c90` 和等效别名
- 新的 `lint` 选项如下：
  - `-features=[no%]gcc_enums`
  - `-fshort-enums`
- 与提高的 gcc 兼容性关联的功能如下：
  - 类型中大小为零的结构
  - `__builtin_offsetof` 内部函数
  - `__builtin_expect` 内部函数
  - 接受续行符（反斜杠 \）后的空格
  - 正确处理 BOM 字符
  - 在 `-fshort-enums` 选项下允许 char 和 short 枚举
  - 通过 `-features=[no%]gcc_enums` 允许 unsigned int 枚举
  - 通过 `-features=[no%]gcc_enums` 允许 long 和 unsigned long 枚举
  - 枚举上允许 packed 属性
- 支持的新属性如下：
  - `__packed__`
  - `section(…)`
- 增加了 `__cpuid` 内部函数
- 增加了对 C11 字符串文字 `u8""`、`u""` 和 `U""` 的支持

有关更多信息，请参见 cc(1) 手册页和《Oracle Developer Studio 12.6: C User's Guide》。

♦ ♦ ♦ 第 4 章

代码分析工具

代码分析工具套件通过检测常见的编码错误（包括内存泄漏和访问违规）来确保应用程序的可靠性和稳定性。这使得开发者可以更快地编写错误更少的更好代码。

本章介绍此 Oracle Developer Studio 发行版中代码分析工具的新增功能和更改的功能，其中包含以下各节：

关于代码分析工具

代码分析工具使用静态、动态和代码覆盖分析来检测许多常见的编码错误（包括内存泄漏和内存访问违规），从而帮助应用程序变得更加可靠。Previse 在编译时执行静态分析，discover 在应用程序运行时执行动态分析，以确定代码质量问题。uncover 工具分析代码覆盖数据以提供有关测试套件未涵盖的函数的信息，并告知如何通过覆盖这些函数受益。

使用代码分析器图形工具或 codean 命令行实用程序可以查看三种类型的分析。分析提供了应用程序漏洞的综合视图，使得您可以提高应用程序的正确性和可靠性。

新增 discover 功能

此发行版的 discover 内存分析工具中增加了以下功能。

- **增强了 discover 针对以前检测过的二进制文件的使用** – discover 实用程序在使用二进制文件的已检测版本覆盖二进制文件之前会自动将其保存。此实用程序现在可以直接运行，不必保存原始二进制文件。此外，discover 可以与 uncover 在同一二进制文件上运行，不必显式保存该二进制文件。对于 Linux 和 Oracle Solaris 上的二进制文件，现在缺省情况下会生成注释。

有关更多信息，请参见《Oracle Developer Studio 12.6: Discover and Uncover User's Guide》和 discover(1) 手册页。

新增 uncover 功能

此发行版的 uncover 代码覆盖工具中增加了以下功能。

- **增强了 uncover 针对以前检测过的二进制文件的使用**－uncover 实用程序在使用二进制文件的已检测版本覆盖二进制文件之前会自动将其保存。现在可以直接对检测过的二进制文件运行此实用程序。此外，uncover 可以与 discover 在同一二进制文件上运行，不必保存该二进制文件。对于 Linux 和 Oracle Solaris 上的二进制文件，现在缺省情况下会生成注释。

有关更多信息，请参见 uncover(1) 手册页和《Oracle Developer Studio 12.6: Discover and Uncover User's Guide》。

♦ ♦ ♦ 第 5 章

5 性能分析工具

性能分析工具配合使用，可供您分析应用程序的行为和找到影响性能的问题症结。

本章介绍此 Oracle Developer Studio 发行版中性能分析工具的新增和更改的功能。

关于性能分析器

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。它可确定哪些函数、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。

性能分析器提供了一组命令和工具，例如：

collect utility	收集针对用户级程序的分析数据。
`er_kernel` utility	收集针对 Oracle Solaris 内核的分析数据。
`er_print` utility	以文本形式提供分析信息。
性能分析器 GUI	以图形形式提供分析信息。

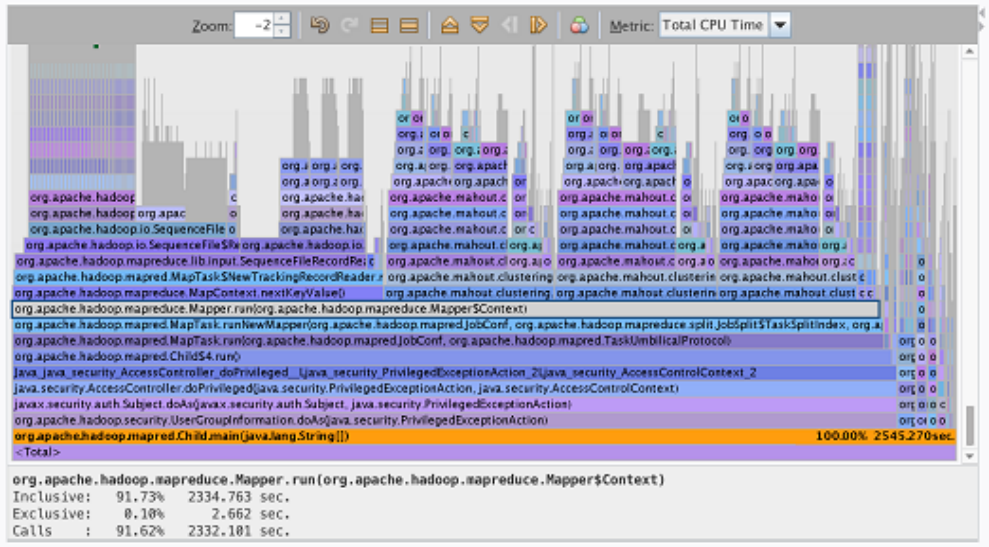
线程分析器是一款可让您专注于多线程问题的相关工具。

有关更多信息，请参阅《Oracle Developer Studio 12.6: Performance Analyzer》和《Oracle Developer Studio 12.6: Performance Analyzer Tutorials》。

性能分析器新功能

本节概述了此发行版中的性能分析器和相关工具的新功能。有关更多信息，请参见性能分析器中的 "Help"（帮助）。

- **新的可视化调用树视图**－新视图 "Visual Call Tree"（可视化调用树）以称作火焰图的图形格式显示热门的执行路径。



有关更多信息，请参见《Oracle Developer Studio 12.6: Performance Analyzer》 中的 “Flame Graph View”。

- **新增了对 Scala 应用程序的支持**－首次增加了对 Scala 应用程序分析的支持。
- **增强了 GNU 内联分析**－对于使用 GNU 编译器编译的二进制文件，PC 视图和 "Disassembly"（反汇编）视图现在可以识别与最末端的内联代码关联的内联指令和源代码。
- **内存空间增强功能**－内存空间视图名称已更新，可以提供用户级视图名称和说明。
- **改进了对 <未记录 Java 调用堆栈> 的分析**－当 JVM 无法捕获给定抽样的当前 Java 调用堆栈时，分析器会将抽样报告为 <未记录 *Java* 调用堆栈>。在此发行版中，PC 视图将描述与缺少的 Java 调用堆栈关联的错误代码。此外，您可以使用过滤器来隔离 <未记录 *Java* 调用堆栈> 并切换到计算机模式来分析缺少的 Java 调用堆栈和函数。
- **大型 Jar 文件支持**－现在支持使用大型 Jar 文件的 Java 和 Scala 应用程序。
- **Intel 编译器 `-ipo` 支持**－性能分析器现在支持使用 Intel `-ipo` 标志编译的代码。
- **视图历史记录中的导航功能**－新控件支持导航到以前打开的视图。

命令行工具的更改

本节介绍各个命令行性能分析工具的更改。有关更多信息，请参见每个命令行工具相应的手册页。

对数据收集工具的更改

数据收集工具包括 collect 命令、dbx collector 命令和 er_kernel 命令。这些工具全都用于分析程序，以收集数据和创建实验，性能分析器或 er_print 可以读取此类数据和实验。

collect 实用程序更改

collect 实用程序是在运行时分析应用程序的工具，可以收集数据和创建实验，且性能分析器或 er_print 可以读取此类数据和实验。

除了所有数据收集工具通用的更改外，本发行版中的 collect 实用程序还有如下更改：

现在可以禁用分析过程中的调用堆栈捕获。在 x86 上对 Java 进行分析时，禁用调用堆栈捕获可以降低发生与堆栈展开相关的致命错误的风险。有关更多详细信息，请参见 collect(1) 中的 SP_COLLECTOR_NATIVE_MAX_STACKDEPTH 信息。

er_kernel 实用程序更改

er_kernel 命令分析 Oracle Solaris 内核，并生成可在性能分析器或 er_print 中检查的实验。

er_kernel 实用程序进行了如下更改：

- 在 OVM (xen) 下运行的 Oracle Solaris VM 上现在支持内核分析。

有关更多信息，请参见 er_kernel(1) 手册页。

er_print 实用程序更改

er_print 实用程序会生成纯文本版本、可由性能分析器显示的数据视图。输出显示在标准输出窗口中。

er_print 实用程序在此发行版中进行了如下更改：

- Oracle Solaris 上提供了命令行编辑功能和历史记录。
- 现在支持使用大型 Jar 文件的 Java 和 Scala 应用程序。
- 性能分析器现在支持使用 Intel -ipo 标志编译的代码。

- 提供了 dumpgc 实用程序，它转储 Java 垃圾收集器事件。
- -pcs 选项现在将显示与缺少的 Java 调用堆栈关联的各种错误代码的说明。
- 对于使用 GNU 编译器编译的二进制文件，PC 视图和 "Disassembly"（反汇编）视图现在可以识别与最末端的内联代码关联的内联指令和源代码。
- 现在提供了对 Scala 应用程序分析的支持。

有关更多信息，请参见 er_print(1) 手册页。

其他命令和 API 的更改

er_archive 命令具有以下更新：

- 新增了命令选项 -d *path*—此选项将内容归档到由绝对路径 *path* 指定的公用目录。
- 新增了命令选项 -r *path*—此选项将内容归档到由相对路径 *path* 指定的公用目录。

不再支持命令 bw、ripc、spot、spot_cmds、spot_cmds_timing、spot_diff 和 traps。

♦ ♦ ♦ 第 6 章

6 调试工具

Oracle Developer Studio 提供了命令行 dbx 调试器以及用于使用 dbx 的 dbxtool 图形工具。调试器还集成到 IDE 中。有关使用 IDE 进行调试的更多信息，请参见第 7 章 Oracle Developer Studio IDE。

本章包含以下主题：

关于 dbx 调试器

dbx 调试器是一个交互式源代码级事后和实时调试工具。可以在命令行通过 dbxtool 图形界面和在 Oracle Developer Studio IDE 中使用该调试器。dbx 调试器是可脚本化并且多线程感知的。

新增和更改的 dbx 功能

以下是 dbx 中添加或更改的功能：

- 运行时检查 (runtime checking, RTC) 现在支持 discover 引擎。
- 向量寄存器能够以不同的格式显示。有关更多信息，请参见 help examine 和 help registers。
- Linux 上现在也支持一个嵌入式 python 解释器和新生插件 API，它主要用于编写美化输出过滤器。

有关更多信息，请参见《Oracle Developer Studio 12.6: Debugging a Program with dbx》、dbx(1) 手册页以及 dbx 帮助文件。要访问 dbx 帮助文件，请键入以下命令：

```
% dbx
(dbx) help
```

有关更多信息，请在 dbx 下发出 `help changes` 命令来访问 dbx 帮助文件。

♦ ♦ ♦ 第 7 章

Oracle Developer Studio IDE

Oracle Developer Studio IDE 为喜欢图形编程环境的用户集成了 Oracle Developer Studio 的许多组件。

本章阐述下列主题：

关于 Oracle Developer Studio IDE

Oracle Developer Studio 提供了图形集成开发环境 (integrated development environment, IDE)，其生成于 NetBeans 平台之上，配置为使用 Oracle Developer Studio C、C++ 和 Fortran 编译器、`dmake` 分布式 make 命令和 `dbx` 调试器。IDE 还与分析套件的一些分析器工具集成，使得您可以在不离开 IDE 的情况下对代码进行分析。

用于启动 IDE 的命令是 `devstudio`。有关更多信息，请参见 `devstudio(1)` 手册页。

有关 IDE 的完整文档，请参见 IDE 中的帮助。有关使用 IDE 基本功能的逐步说明，请参见《Oracle Developer Studio 12.6: IDE Quick Start Tutorial》。

新增和已更改的 IDE 功能

以下是 Oracle Developer Studio IDE 中添加或更改的功能：

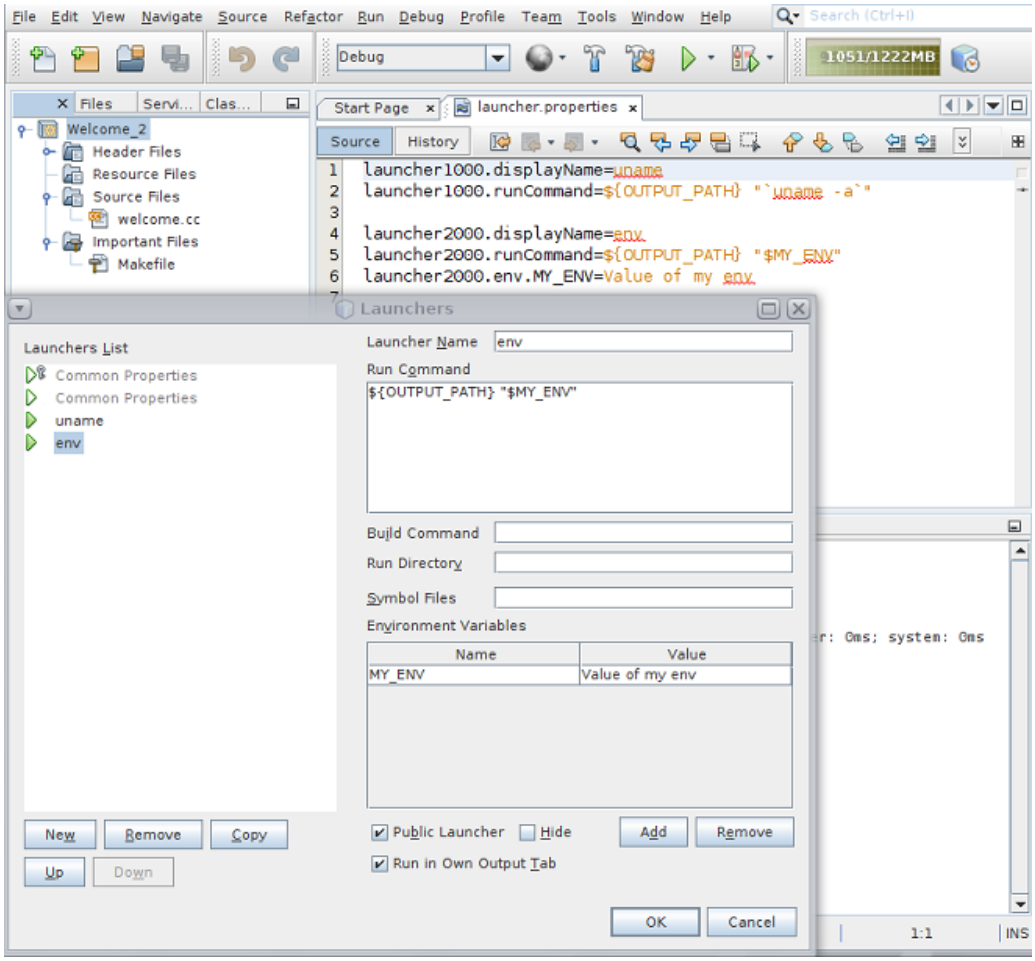
- **管理启动器的运行/调试配置**－有关更多信息，请参见“用于运行/调试启动器的 UI” [28]。
- **修改 PATH 变量**－有关更多信息，请参见“前置或附加工具链路径” [29]。
- **文本字段中的环境变量**－有关更多信息，请参见“添加环境变量” [31]。

- **基于工具集合包装器构建分析器**－有关更多信息，请参见“构建分析器” [32]
- **清除 C/C++ 高速缓存**－有关更多信息，请参见“清除 C/C++ 高速缓存” [34]
- **在“查找使用实例”中区分读取/写入访问权限**－有关更多信息，请参见“区分读取/写入访问权限” [34]
- **终端扩展**－有关更多信息，请参见“终端功能扩展” [35]。
- **新的可固定监视**－有关更多信息，请参见“可固定监视” [37]

用于运行/调试启动器的 UI

可以使用新的 UI 管理运行/调试命令配置。要打开配置窗口，请从项目的上下文菜单中单击 *Manage Launchers*（管理启动器）。

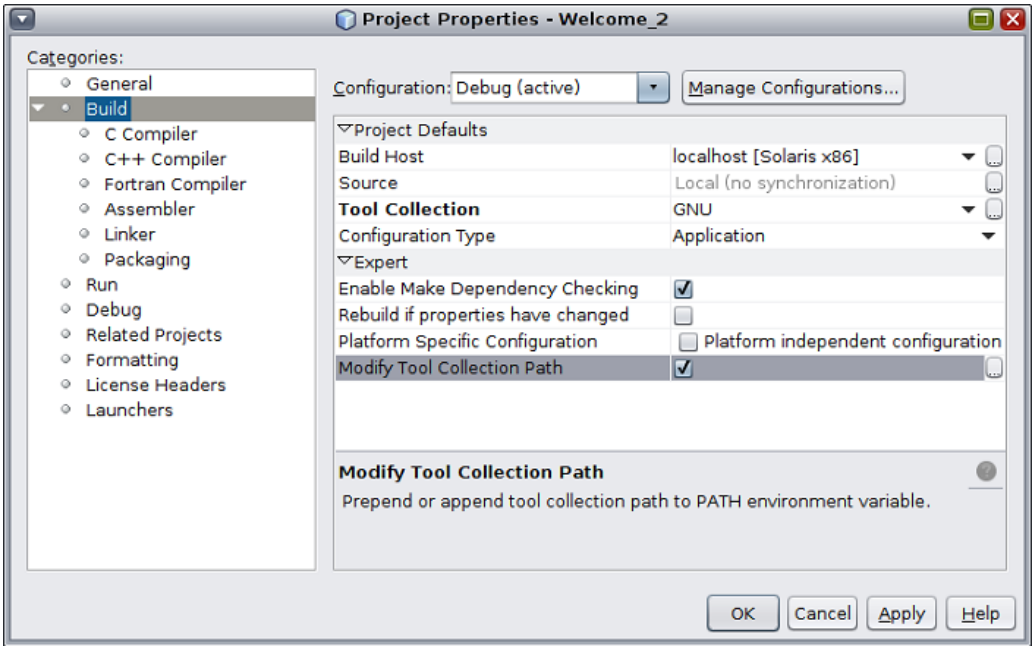
图 2 用于运行/调试启动器的 UI



前置或附加工具链路径

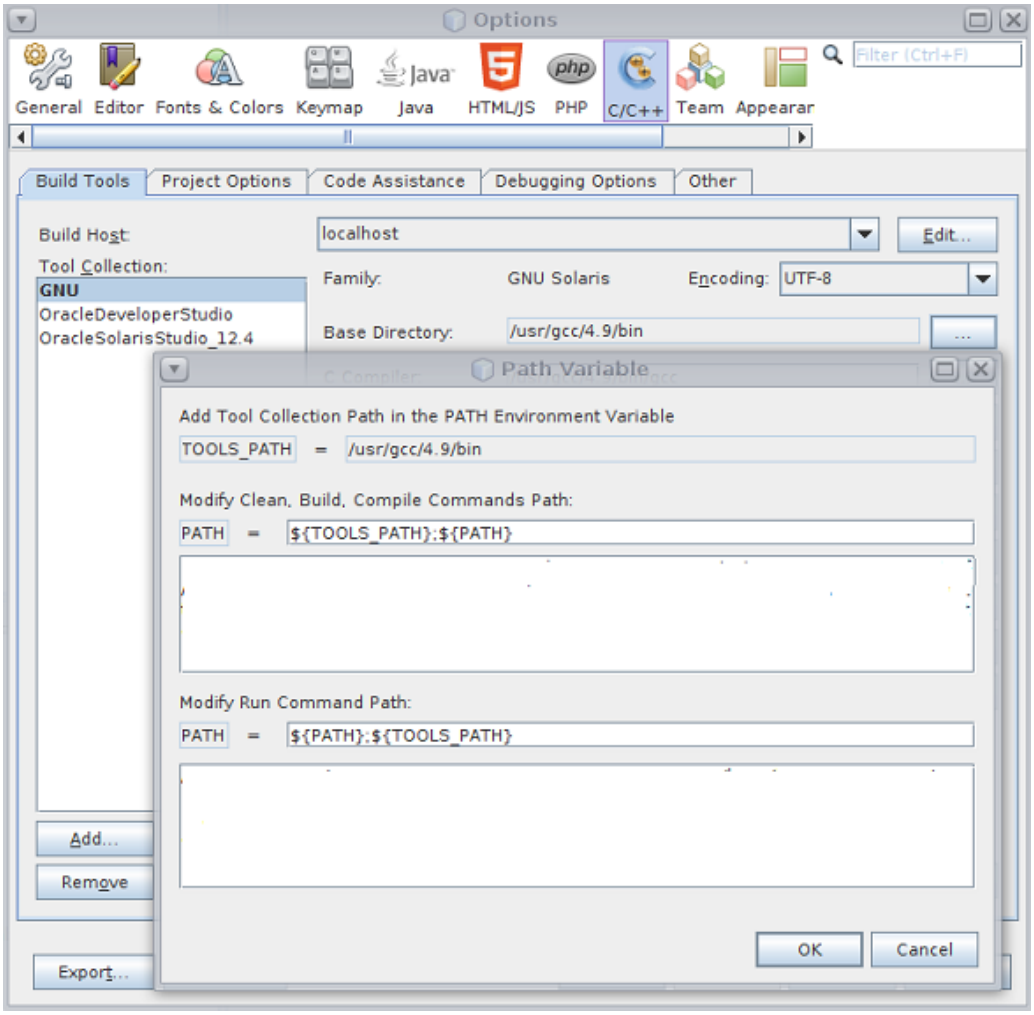
Project Properties（项目属性）下的一个新选项允许您指定是否应当将工具链目录附加到 PATH 变量。

图 3 "Modify Tool Collection Path"（修改工具集合路径）选项



缺省情况下将前置 `$TOOLS_PATH`。但是，可以在 `Options`（选项）窗口中修改此设置。

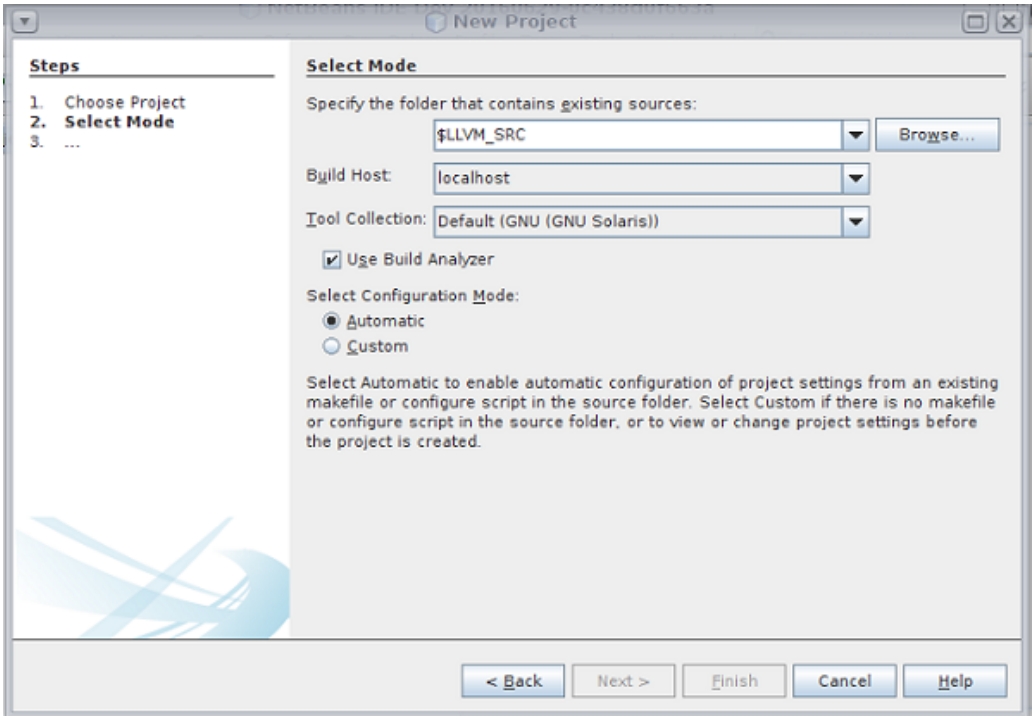
图 4 修改工具集合路径



添加环境变量

在使用现有源代码或基于二进制文件创建项目时，可以使用环境变量。

图 5 添加环境变量



构建分析器

IDE 在 Windows 和 Mac 上创建了工具集合包装器。此包装器用来构建项目，因此，将存储所有编译命令行并将其用于配置代码帮助。

图 6 构建分析器

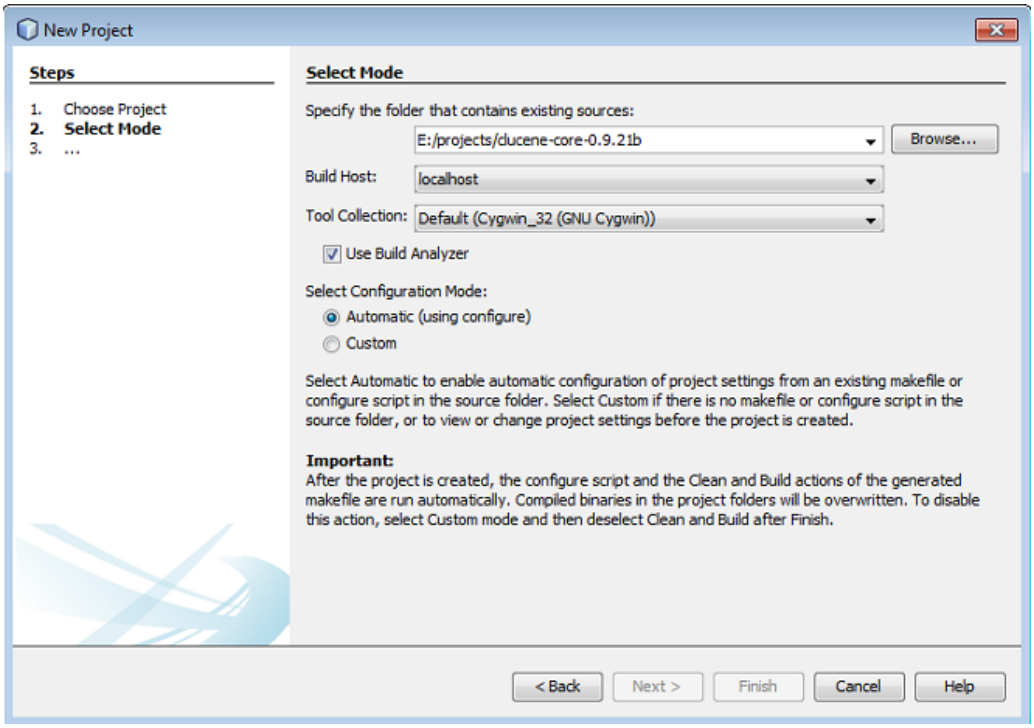
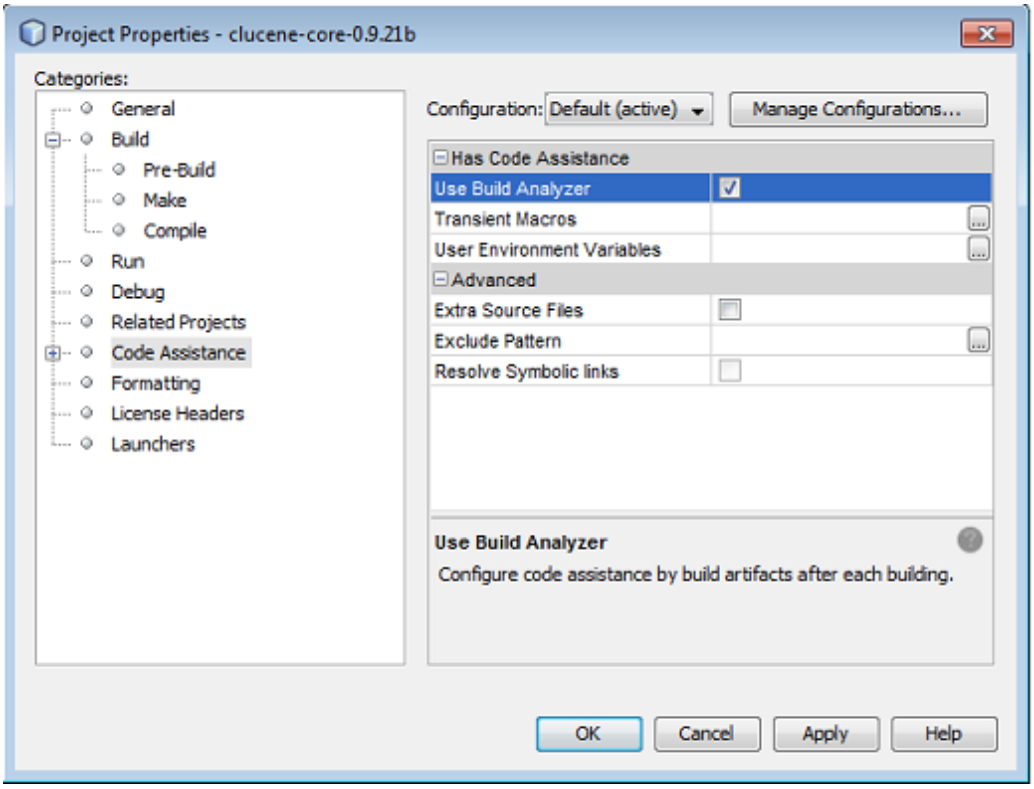

New Project
Steps
1. Choose Project
2. Select Mode
3. ...
Select Mode
Specify the folder that contains existing sources:
E:/projects/clucene-core-0.9.21b
Browse...
Build Host:
localhost
Tool Collection:
Default (Cygwin_32 (GNU Cygwin))
Use Build Analyzer
Select Configuration Mode:
Automatic (using configure)
Custom
Select Automatic to enable automatic configuration of project settings from an existing makefile or configure script in the source folder. Select Custom if there is no makefile or configure script in the source folder, or to view or change project settings before the project is created.
Important:
After the project is created, the configure script and the Clean and Build actions of the generated makefile are run automatically. Compiled binaries in the project folders will be overwritten. To disable this action, select Custom mode and then deselect Clean and Build after Finish.
< Back
Next >
Finish
Cancel
Help

图 7 配置代码帮助



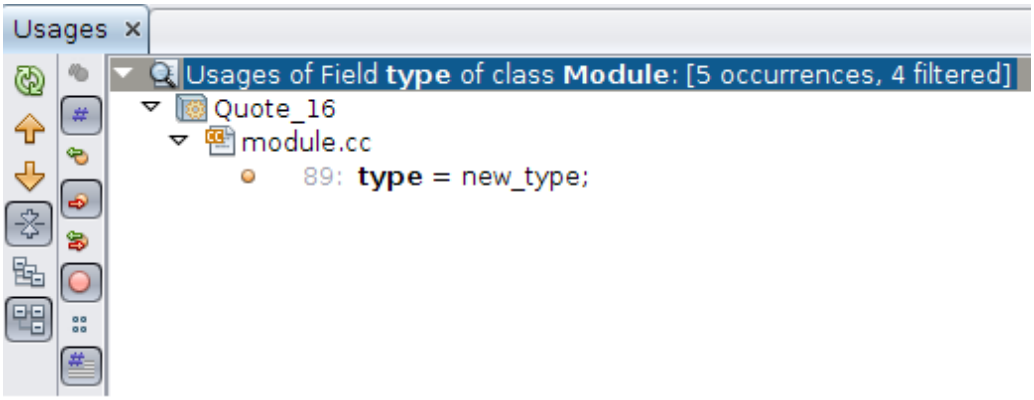
清除 C/C++ 高速缓存

Code Assistance（代码帮助）下添加了一个新选项 *Clean C/C++ cache and restart IDE*（清除 *C/C++* 高速缓存并重新启动 *IDE*）。可以使用此选项清除高速缓存目录并重新启动 IDE。

区分读取/写入访问权限

可以根据提供给变量的访问权限对变量进行标记。具有读取访问权限、写入访问权限或读取/写入访问权限的变量分别以不同的方式显示。

图 8 区分读取/写入访问权限



终端功能扩展

添加了以下终端功能扩展。

- **从终端在编辑器中打开文件**－选择 *Window*（窗口）-> *IDE Tools*（*IDE* 工具）-> *Terminal*（终端）并导航到 "Project"（项目）文件夹。可以键入以下命令在编辑器中打开指定的文件。
  - 要打开单个文件，请键入：

    ```
    ideopen module.cc
    ```

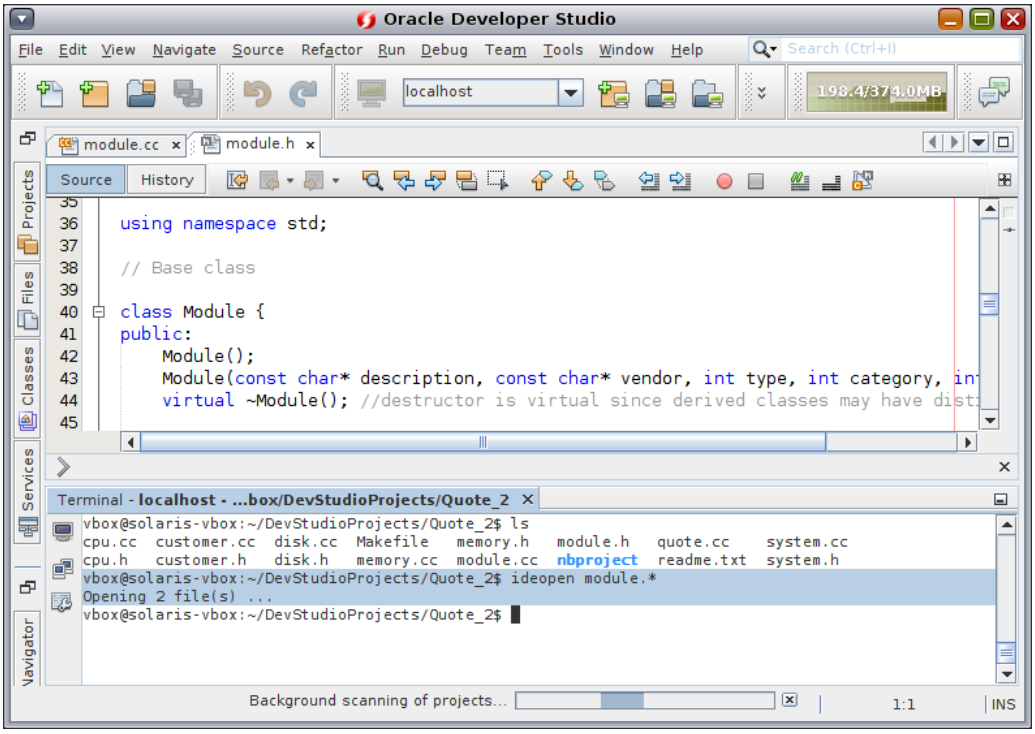
  - 要打开多个文件，请键入：

    ```
    ideopen module.cc module.h
    ```

  - 要使用（您的当前 shell 支持的）通配符打开多个文件，请键入：

    ```
    ideopen module.*
    ```

图 9 在编辑器中打开文件

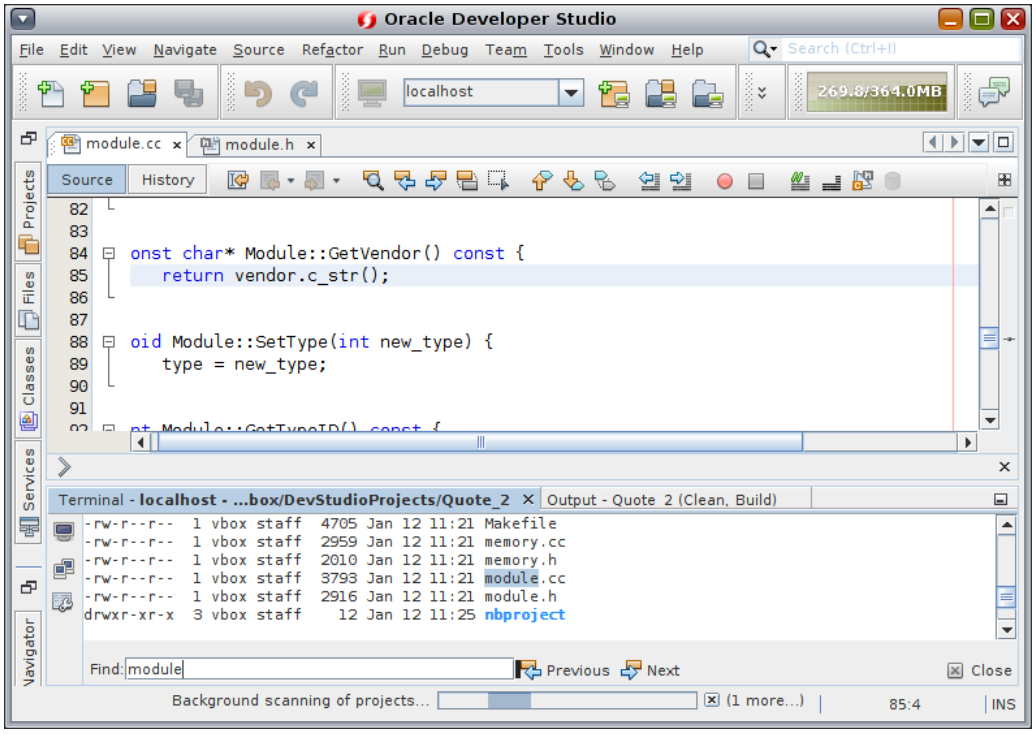


- **支持键盘快捷键**－此功能使得用户可以使用键盘打开新的选项卡以及在打开的选项卡之间切换。使用 *Ctrl+Alt+Shift+T* 可以打开新的选项卡，使用 *Alt+1*、*Alt+2* ... 可以在选项卡之间切换。
- **从终端访问超链接**－如果程序的输出在终端中输出了某个文件的链接，并且用户在终端中单击了此链接，则 IDE 将在编辑器中打开此文件。可以使用转义序列在终端中输出超链接。例如：

  ```
  fprintf(stdout, "(\033]10;%s;%s\007)]\n", "/home/ilia/NetBeansProjects/
  CppApplication_48/main.cpp:20", "main.cpp:20")
  ```

- **在终端中进行搜索**－终端中现在实现了搜索功能。要进行搜索，请使用 *Ctrl+Shift +F*，或者 *[*单击右键*]*并选择 *Find in Terminal*（在终端中查找）。还可以在找到的实例之间进行导航。使用 *Shift + F3* 可以导航到前一实例，使用 *(F3, Enter)* 可以导航到下一实例。

图 10 在终端中进行搜索

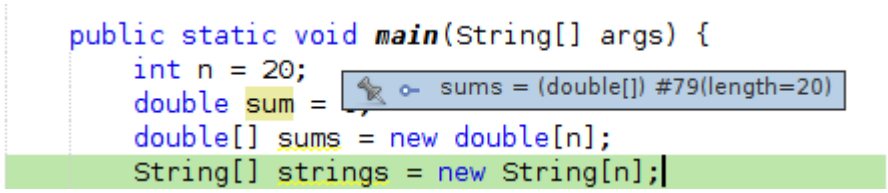


可固定监视

监视现在可以固定到编辑器区域，将鼠标指针悬停在变量或选定项上时，将会显示工具提示及其值。

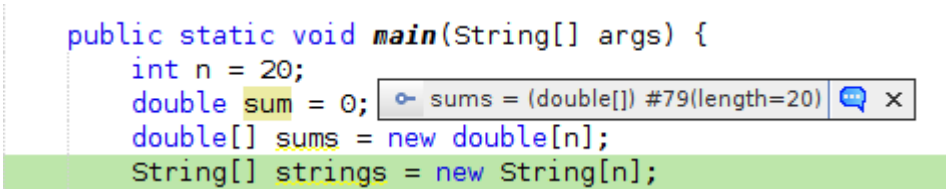
工具提示现在包含一个大头针图标，单击该图标时，将会创建一个固定到编辑器的监视。

图 11 工具提示中的大头针图标



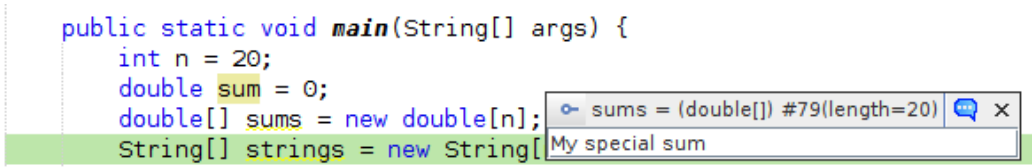
将显示大头针监视窗口而非工具提示。可以通过使用鼠标进行拖动来调整此窗口的位置。

图 12 大头针监视窗口



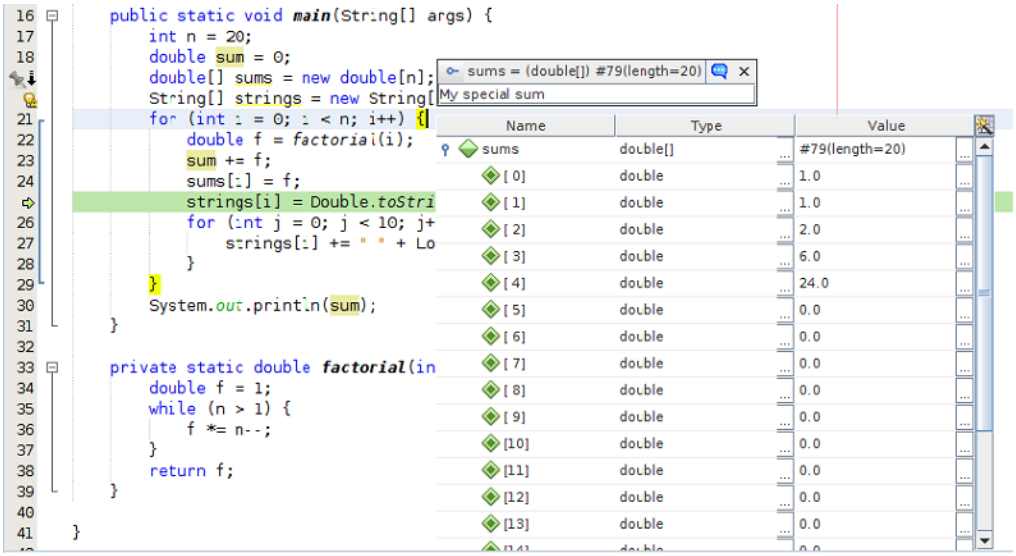
大头针监视窗口在右侧包含两个图标：注释图标和关闭图标。单击注释图标将显示一个文本字段，可以在其中添加关于监视的注释。

图 13 注释图标和关闭图标



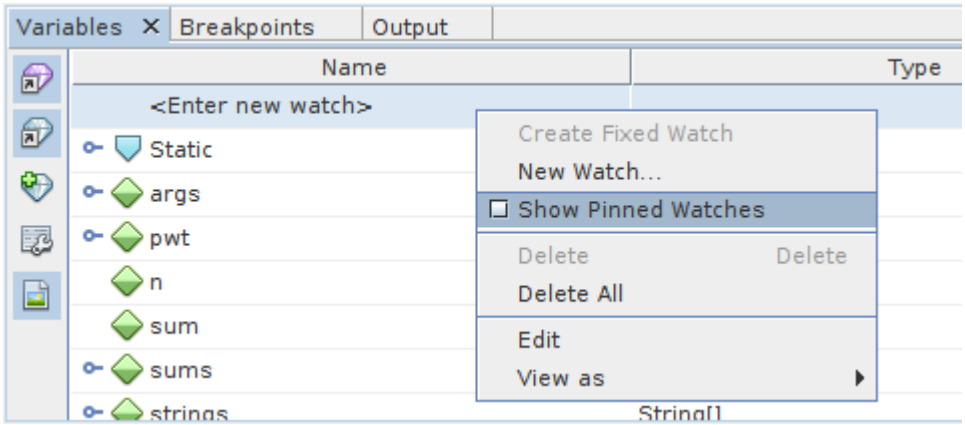
当大头针监视显示结构化值时，它将在左侧显示一个展开图标。当展开时，将显示包含其子代的视图：

图 14 展开图标



缺省情况下，*Variables/Watches*（变量/监视）窗口不显示已固定的监视。但是，可以使用 *Show Pinned Watches*（显示固定的监视）上下文操作来查看已固定的监视。

图 15 "Show Pinned Watches"（显示固定的监视）选项



Docker

Oracle Developer Studio 为用户提供了 Docker 支持。Oracle Developer Studio IDE 针对 Docker 文件提供了突出显示和自动完成功能，这使得用户能够轻松与 Docker 进行交互。用户可以连接到在其计算机上或在远程计算机上运行的 Docker 守护进程。用户可以使用 IDE 从 Docker Hub 下载映像，构建其自己的映像，以及运行和管理 Docker 容器。

关于 Docker

Docker 是一个开源容器引擎，用于创建、执行和管理轻量级应用程序容器。Docker 允许用户运行通过公共系统信息库获得的已发布映像中的实例。还可以使用一种特定于域的语言（称为 Dockerfile）来创建新映像。在创建后，映像可以在任何 Docker 环境中运行。

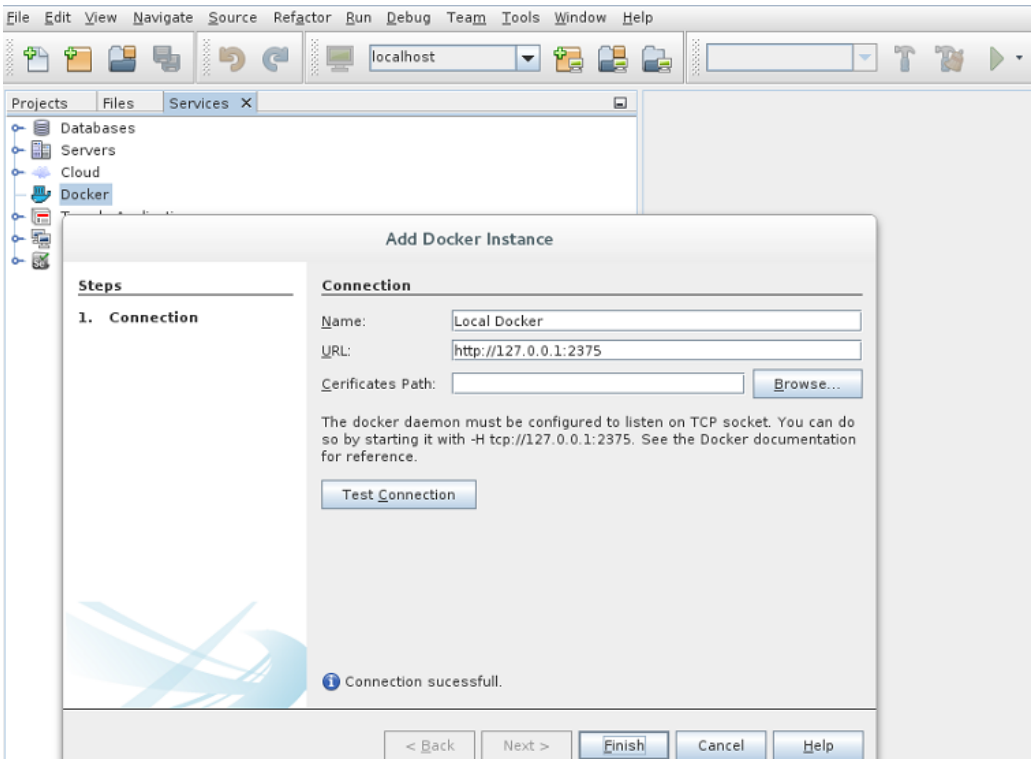
与 Docker 进行交互

本节介绍了使用 Oracle Developer Studio IDE 与 Docker 进行交互的步骤。

▼ 如何使用 Oracle Developer Studio IDE 与 Docker 进行交互。

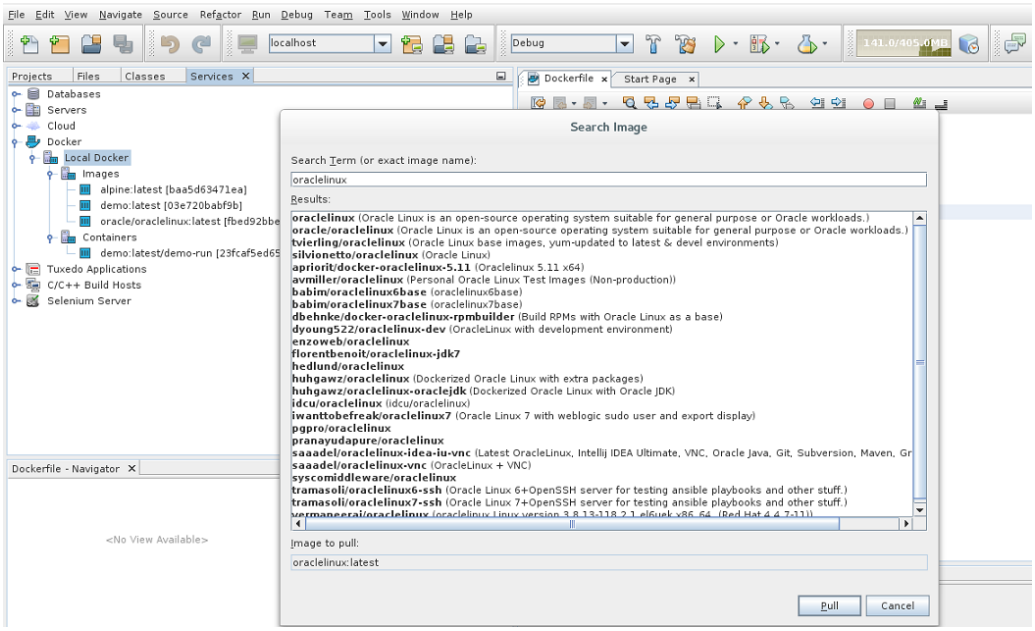
用户可以对在其本地计算机上或在远程计算机上运行的 Docker 守护进程执行以下任务。不过，与在远程计算机上运行的 Docker 守护进程进行交互提供了将 Oracle Developer Studio IDE 用于云环境的其他机会。

1. **使用 `http` 连接到 Docker 守护进程。**

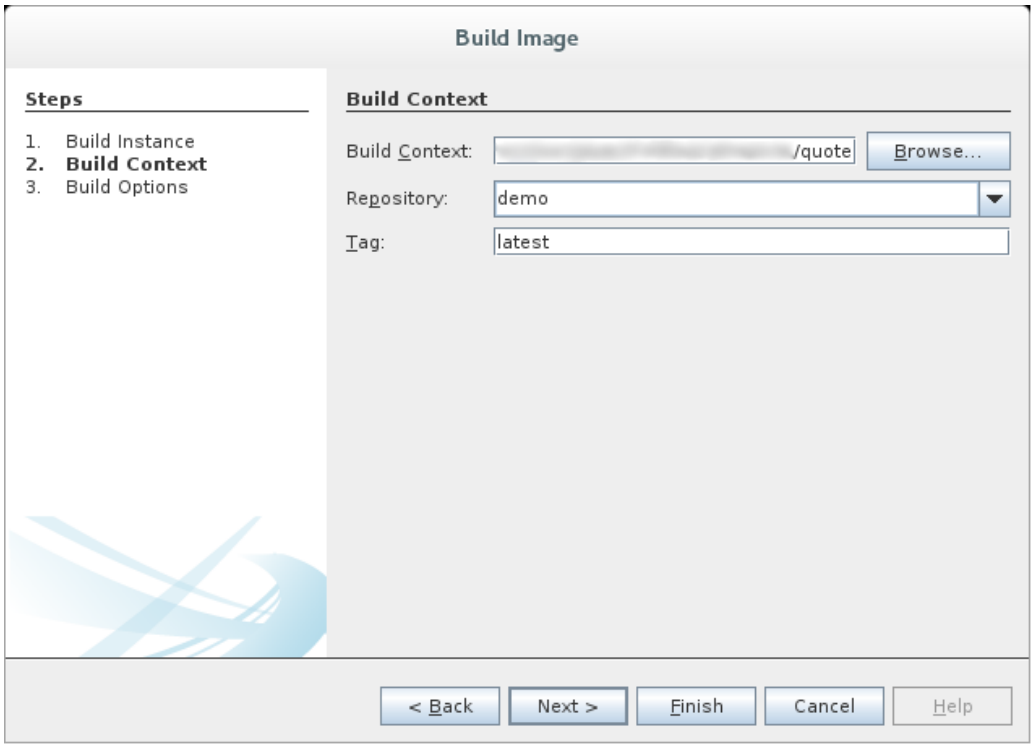


分别在 `Name`（名称）和 `URL` 字段下指定 Docker 的名称和 URL，然后单击 *Finish*（完成）。

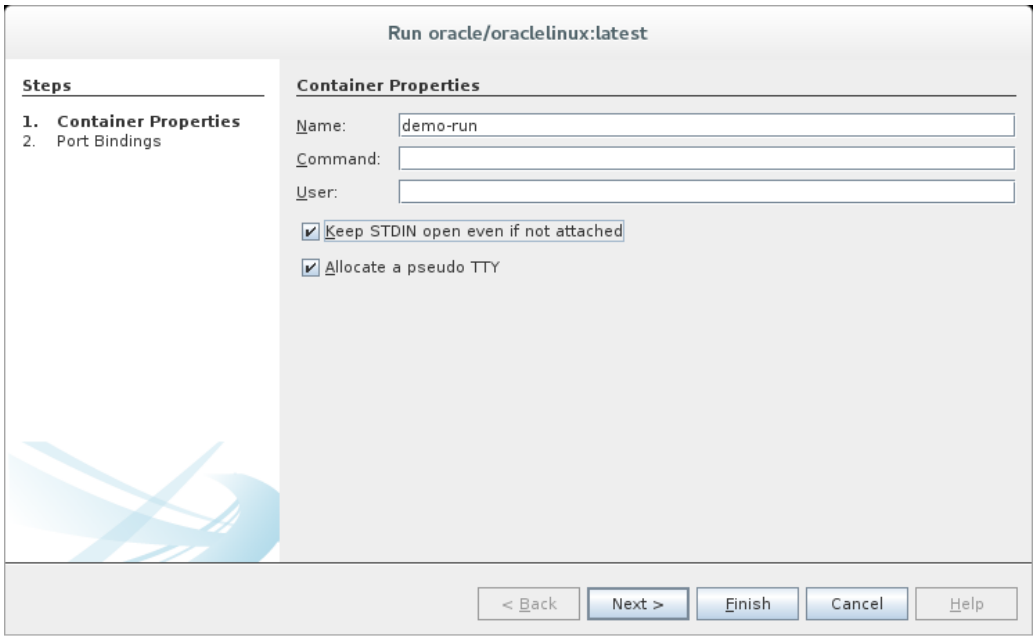
2. **在 *Services*（服务）视图中，将会显示新节点，可以使用该节点从 Docker Hub 中搜索和下载任何映像。**



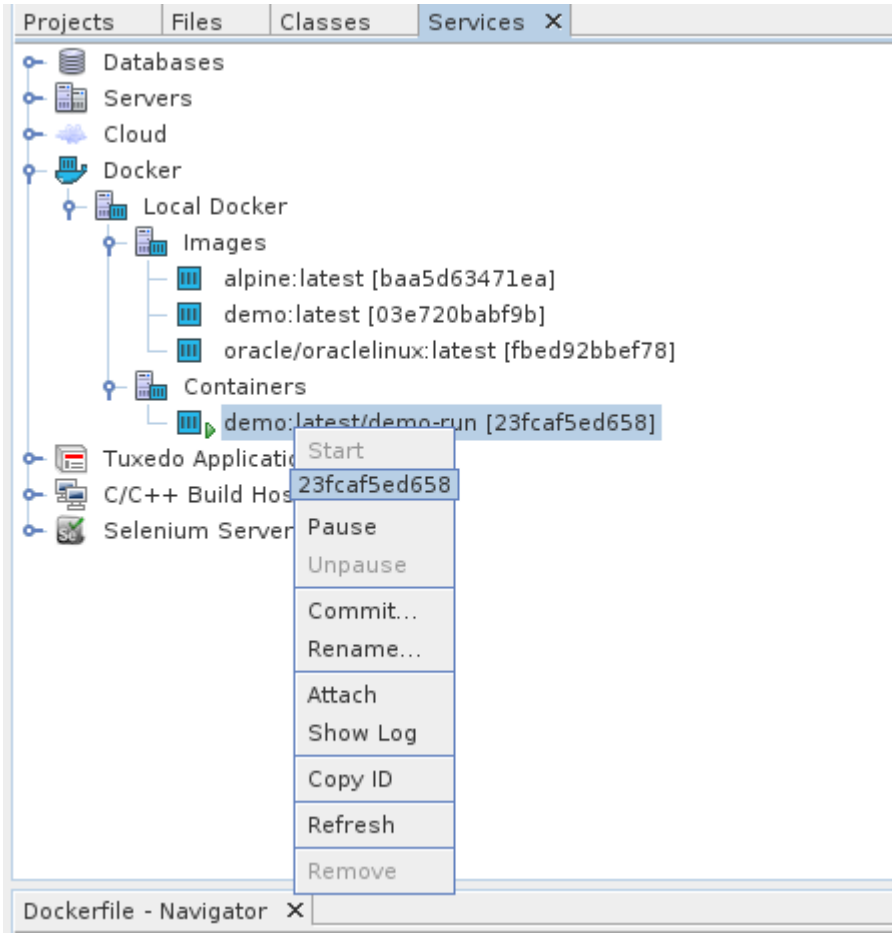
3. **在 *Projects*（项目）视图中，右键单击 Dockerfile，然后单击 *Build*（构建）来构建映像。在 *Build Image*（构建映像）窗口中，选择 *Build Context*（构建上下文）并输入详细信息。**



4. **运行 Docker 容器。使用 IDE 终端与容器进行交互。**



5. **在 *Services*（服务）视图中，右键单击容器来管理容器。**



♦ ♦ ♦ 第 8 章

8 OpenMP API

本章介绍此 Oracle Developer Studio 发行版中 OpenMP API 支持的更改。

OpenMP

本节讨论 OpenMP API 的新增功能和更新。

- 改进的动态循环调度－OpenMP 4.5 规范澄清了动态循环调度如何将块分配给线程。循环迭代的块不能保证按单调顺序分配给线程。例如，在下面的 OpenMP 动态循环中，第 50 个迭代在第 0 个迭代之前执行。

  ```
  #pragma omp parallel for num_threads(2) schedule(dynamic)
  for (i=0;i<100;i++) A[i] = B[i];
  ```

  这可以实现更高效的动态循环调度实施。
- OpenMP 任务 Pragma 的优先级子句－优先级子句功能包括以下内容：
  - `OMP_MAX_TASK_PRIORITY` 环境变量
  - `omp_get_max_task_priority()` API 例程

  有关更多信息，请参见 **OpenMP 4.5 规范**第 `2.9.1` 部分中用于任务构造的优先级子句，第 `4.14` 部分中的 `OMP_MAX_TASK_PRIORITY` 环境变量，以及第 `3.2.36` 部分中的 `omp_get_max_task_priority` API 例程。
- 针对线程关联（处理器绑定）的查询函数－添加了以下查询 API 例程：
  - `omp_get_num_places()`
  - `omp_get_place_num_procs()`
  - `omp_get_place_proc_ids()`
  - `omp_get_place_num()`
  - `omp_get_partition_num_places()`
  - `omp_get_partition_place_nums()`

  有关更多信息，请参见 **OpenMP 4.5 规范**第 256 页的第 `3.2.23` 部分和第 261 页的第 `3.2.28` 部分。

有关 OpenMP 的更多信息，请参见《Oracle Developer Studio 12.6: OpenMP API User's Guide》。

♦ ♦ ♦ 第 9 章

9 其他更改

本章介绍 Oracle Developer Studio 软件的其他组件的新增和更改的功能。

编译器中的更改

以下部分介绍编译器中的更改，包括以下主题：

编译器通用的新增和更改的功能

下面列出了自上一发行版起对 C、C++ 和 Fortran 编译器进行的更改。可在编译器手册页中找到详细信息。特定于 C++ 编译器的更改已在第 2 章 C++ 编译器中进行了详细介绍。特定于 C 编译器的更改已在第 3 章 C 编译器中进行了详细介绍。

GCC 兼容选项 `-fvisibility`

`-fvisibility=v` 选项等效于 `-xldscope` 选项，如下所示：

`-fvisibility` 选项	等效的 **`-xldscope`** 选项
`default`	`global`
`internal`	`hidden`
`protected`	`symbolic`

-fvisibility 选项	等效的 -xldscope 选项
hidden	hidden

GCC 兼容选项 -shared

在 Oracle Developer Studio 12.6 中，-shared 选项与 GCC 兼容。-shared 选项创建共享库，类似于 -G 选项。缺省情况下，-shared 选项将链接在创建可执行程序时将链接的同一批库，而 -G 选项不会自动链接缺省库。在以前的发行版中，-shared 选项等效于 -G 选项。

新命令选项

下面的列表介绍了所有编译器共有的新命令选项。

- **适用于 SPARC M8/T8 的新选项**－-xchip 和 -xtarget 值现在可用于 SPARC M8/T8 系统。
- **适用于 x86 处理器 Skylake 的新选项**－-xchip=skylake、-xtarget=skylake 和 -xarch=avx512 值现在可用于 Intel Skylake 处理器。
- **Linux 上的缺省选项**－Linux 上的缺省选项是 -xannotate。
- 命令选项 -fstrict-aliasing 和 -fno-strict-aliasing 现在可用。

Oracle Solaris 的新宏

新宏 __SUNOS_Release 现在仅可用于 Oracle Solaris。

Oracle Solaris 上的 __SunOS_RELEASE

一个十六进制值 0xRRrrmm，表示 Oracle Solaris 发行版，其中 RR.rr 是 sysinfo (SI_RELEASE) 系统调用或 uname -r 命令的输出，必要时添加前导零。mm 位保留供将来可能推出的宏发行版使用。所有数位都是十进制的。

例如，Oracle Solaris 11 是 SunOS 5.11，因此 __SunOS_RELEASE 具有值 0x051100。

较旧 Oracle Solaris 发行版的 __SunOS_RELEASE 值始终小于更高发行版的值。例如，

```
#if __SunOS_RELEASE >= 0x051100 // Solaris 11 or later
```

编译器的内联行为

在 -O3 级别，Studio 编译器会自动内联其正文小于调用方开销的例程。要控制自动内联的函数，请使用 -xinline=list 选项。

压缩调试节

可以通过 `-xcompress=debug` 选项使用由 `-xcompress_format` 选项指定的格式来压缩调试节。还添加了对等效 gcc 选项 `-gz` 的支持。

`-xdebugformat=stabs` 选项

`-xdebugformat=stabs` 选项已删除。

Fortran 编译器

Fortran 编译器通过针对 Fortran77、Fortran90 和 Fortran95 标准的创纪录运行时性能和兼容性选项来支持技术和科学应用程序开发。包括大多数 Fortran 2003 功能和 OpenMP 4.5 支持。Fortran 编译器与 C 和 C++ 编译器使用相同的高性能代码生成技术，从而确保结果应用程序为最新的基于 SPARC 和 x86 的 Oracle 系统生成最高性能的并行代码。

Fortran 编译器的更改包括“编译器通用的新增和更改的功能” [49]中介绍的更改，以及以下更改：

- 不再支持 Fortran 全局程序检查 (global program checking, GPC)。
- 现在支持 Fortran 2003 标准中引入的参数化派生类型。

有关更多信息，请参见 `f95(1)` 手册页和《Oracle Developer Studio 12.6: Fortran User’s Guide》。

性能库更改

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 性能库以主要来自 `http://www.netlib.org` 上的 Netlib 的公共域子例程集合为基础。Oracle 已增强了这些公共域子例程并将其捆绑为 Oracle Developer Studio 性能库。

对于此发行版，Oracle Developer Studio 性能库中的 LAPACK 已升级为版本 3.6.1。Oracle Developer Studio 性能库中实现了 LAPACK 3.6.1 中的所有新功能。有关更改的更多信息，请参见 `http://www.netlib.org/lapack/lapack-3.6.1.html`。

有关更多信息，请参见《Oracle Developer Studio 12.6: Performance Library User’s Guide》。

索引