Oracle® Solaris Studio 12.4：发行说明

文件号码 E57203
2015 年 5 月

ORACLE

文件号码 E57203

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** – 提供系统要求以及有关此 Oracle Solaris Studio 12.4 发行版的已知问题及其他问题的信息。
- **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

有关编译器和工具的更多信息，请参见相关手册页或 "Help"（帮助）以及 http://docs.oracle.com/cd/E37069_01 中的文档库。

反馈

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

Oracle Solaris Studio 12.4 发行说明

此文档提供在安装和使用产品之前需要了解的信息。有关安装说明，请参见《Oracle Solaris Studio 12.4：安装指南 》。

系统要求

Oracle Solaris Studio 12.4 软件可以安装在基于 SPARC 或基于 x86 的平台上的 Oracle Solaris 10 或 Oracle Solaris 11.2 操作系统中，也可以安装在 Oracle Linux 操作系统中。

表 1 系统要求

系统资源	要求
硬件	64 位 SPARC 和 x86 平台
操作系统	Oracle Solaris 10 8/11 或 Oracle Solaris 10 1/13 Oracle Solaris 11.2 Oracle Linux 5.8–5.10、6.0–6.5 Red Hat Linux 5.8–5.10、6.0–6.5 Oracle Unbreakable Enterprise Kernel (UEK) 2 和 3 请参见"要求的系统软件包" [10]以确保您的系统具有所需的软件包。
分发	Oracle Solaris 10 的 SVR4 软件包安装 Oracle Solaris 11.2 的 IPS 软件包安装 Oracle Linux 的 RPM 软件包安装 所有平台的 tar 文件 有关所有分发的安装说明，请参见《Oracle Solaris Studio 12.4：安装指南 》。
VM 支持	在 Solaris 10 或 Solaris 11 上作为非全局区域、LDOM 或内核区域运行 Oracle Solaris Studio 的 Oracle Solaris 11.2 全局区域
支持的客户机系统 用于安装分发的 IDE 或性能分析器供远程使用	运行 IDE、性能分析器、代码分析器和 `dbxtool` 的客户机系统必须安装要求的最低 Java 版本

系统资源	要求
Java 版本	Java 7，最低版本 1.7.0_25 Java 8

要求的系统软件包

在安装 Oracle Solaris Studio 12.4 发行版之前，确认要安装软件的系统具有成功运行 Oracle Solaris Studio 软件所需的系统软件包。下表列出了每个支持的操作系统所需的软件包。

表 2 Oracle Solaris Studio 所需的系统软件

操作系统版本	系统软件包
Oracle Solaris 10 8/11 Oracle Solaris 10 1/13	SUNWhea SUNWarc SUNWcpcu SUNWcpp SUNWcsl SUNWlibC SUNWlibm SUNWlibms SUNWpiclh SUNWpiclr SUNWpiclu SUNWsprot SUNWlibstdcxx4 SUNWPython SUNWPython-devel
Oracle Solaris 11.2	有关更新 pkg://solaris/consolidation/sunpro/sunpro-incorporation 的重要信息，请参见《Oracle Solaris Studio 12.4：安装指南 》中的“更新 Oracle Solaris Studio 12.4 必需的 Oracle Solaris 11 系统库”。 pkg://solaris/consolidation/sunpro/sunpro-incorporation@0.5.11-0.175.2.1.0.4.0 pkg:/developer/base-developer-utilities pkg:/developer/library/lint pkg:/developer/library/xprofile pkg:/diagnostic/cpu-counters pkg:/library/glib2 pkg:/library/libxml2 pkg:/library/unixodbc pkg:/library/zlib pkg:/runtime/perl-512 pkg:/shell/bash pkg:/shell/ksh pkg:/system/header pkg:/system/library/c++-runtime pkg:/system/library/math pkg:/system/library/openmp pkg:/system/linker pkg:/system/resource-mgmt/resource-pools
Oracle Linux 5.8 - 5.10	开发/库软件包组和 32 位运行时支持库，包括： glibc glibc.i686

操作系统版本	系统软件包
Oracle Linux 6.0 - 6.5	glibc-devel glibc-devel.i686 elfutils-libelf-devel elfutils-libelf-devel.i686 zlib zlib.i686 libstdc++ libstdc++.i686 libgcc libgcc.i686

安装说明

有关所有分发的安装说明，请参见《Oracle Solaris Studio 12.4：安装指南》。

此发行版中已经删除的功能

Oracle Solaris Studio 12.4 软件中删除了以下功能。

DLight 分析工具

DLight 是一种图形分析工具，在 Oracle Solaris Studio 12.3 和 12.2 中提供。可以使用 Oracle Solaris Studio IDE 通过 DLight 中的部分相同工具来分析项目。

也可以通过现在更容易使用的性能分析器来分析应用程序。

C++ Rogue Wave `tools.h++` 库

`tools.h++` 是一种 C++ 基础类库，在以前的 Oracle Solaris Studio 发行版和 Sun Studio 软件中提供。该库于 1996 年发布，并且此后没有重大更新。时间和日期类具有无法修复的严重问题。

`tools.h++` 中的功能在 Oracle Solaris Studio 中提供的 C++ 标准库中以及在开源的 BOOST 库中随不同的编程接口 (API) 提供。有关 BOOST 库的信息，请参见 `http://www.boost.org`。

`systent.h` 头文件

此 C++ 头文件是从 Cfront 留下来的，并且早于 `unistd.h`。改用 `unistd.h`。

旧的 Fortran 模块格式

由 Forte Developer 7 Fortran 95 7.0 及早期发行版生成的模块格式将不再受支持。本发行版中的 Fortran 编译器无法读取使用较旧的发行版创建的 `.mod` 文件。

编译器选项 `-xbinopt=prepare`

`-xbinopt=prepare` 选项已过时。应改用 `-xannotate=yes`。

编译器选项 `-xcrossfile`

`-xcrossfile` 选项已过时。应改用 `-xipo` 选项。

汇编程序 `-xarch` 选项

不再支持 Oracle Solaris x86 汇编程序 `fbe` 命令选项 `-xarch=amd64` 和 `-xarch=generic64`。

应改用 `-m64` 选项。

注册功能

Sun Studio 用户用于向 Sun Microsystems 注册以及接收最新产品信息的注册工具已从 2011 年开始废弃。该功能已从 Oracle Solaris Studio 产品中删除。

未来发行版中可能会删除的功能

未来的发行版中可能会删除以下功能。

编译器 `-xarch` 选项

对于所有 Oracle Solaris Studio 编译器，命令选项 `-xarch=amd64` 和 `-xarch=generic64` 均已过时并可能会在未来发行版中删除。

应改用 `-m64` 选项。

C 编译器选项 -Xs

-Xs 选项可能会在未来发行版中删除。如果 C 代码必须使用 -Xs 才能正确构建和编译，应迁移该代码，使其至少符合 ISO C 标准的 C99 版本 (dialect)。换而言之，应能够使用 -std=c99 选项进行编译。

lint 实用程序的 -Nlevel 选项

lint 实用程序中用于指定增强分析级别的 -Nlevel 选项可能会在未来发行版中删除。

lock_lint 实用程序

用于分析多线程程序中锁的 lock_lint 实用程序已过时并可能会在未来发行版中删除。

libgc 实用程序

Studio 中的 libgc 库是一个过时的垃圾收集库并可能会在未来发行版中删除。您应使用 Oracle Solaris libgc 库。

性能库中含可选参数的 F95 接口

libsunperf F95 接口的可选参数功能可能会在未来发行版中删除。不含可选参数的 F95 接口将继续受支持。

VMS 可兼容库 libV77

libV77 库包含与 Fortran 90 内部例程冲突的两个例程，且这两个例程不符合 4 位年份的 Y2K 要求。它们只是为了满足 VMS 兼容性而提供的可选例程，并可能会在未来发行版中删除。

编译器选项 -xanalyze=code

-xanalyze=code 选项已废弃且可能会在未来发行版中删除。您应使用 -xprevise 选项进行编译，而不是生成可使用代码分析器查看的源代码静态分析。

libsunmath 的静态版本

Studio 数学库的静态版本 `libsunmath.a` 可能会在未来发行版中删除。动态版本 `libsunmath.so` 将继续可用。通过将选项 `-Bstatic -lsunmath` 替换为 `-Bdynamic -lsunmath` 选项，可将使用的所有静态库都替换为动态库。

传统 C++ iostream

传统 iostream (`libiostream`) 是 iostream 最初的 1986 版本，其在 1998 C++ 标准中被替换。该库是通过 `-library=iostream` 选项指定的。传统的 iostream 不是一项标准，因此该库没有两种相同的实现方式，使用它的代码不可移植，并且与 C++ 标准库不兼容。请使用 C++ 标准库提供的 iostream 函数。

基于 SPARC 的传统系统的编译器选项

未来发行版中可能会删除对传统系统的支持。其中包括基于 UltraSPARC I、II、IIe、III、IIIi、III+、IV 和 IV+ 处理器的系统。

相应地，可能会删除 `cc`、`CC` 和 `f95` 编译器命令的以下选项：

```
-xchip={ultra,ultra2,ultra2i,ultra2e,ultra3,
        ultra3i,ultra3cu,ultra4,ultra4plus}

-xtarget={ultra,ultra1/140,ultra1/170,ultra1/200,
          ultra2,ultra2/1170,ultra2/1200,ultra2/1300,ultra2/2170,
          ultra2/2200,ultra2/2300,ultra2e,ultra2i,ultra3,ultra3cu,
          ultra3i,ultra4,ultra4plus}
```

编译器选项 -xdebugformat=stabs

未来发行版中可能会删除所有编译器的 `-xdebugformat=stabs` 选项。唯一的调试器格式选项为 `-xdebugformat=dwarf`，这是当前的缺省值。

dbx 中的运行时检查

可能删除 `dbx` 中的运行时检查 (Runtime Checking, RTC) 功能。可以使用 `discover` 工具进行运行时内存检查。

传统 Fortran F77 对象文件

可能停止支持使用传统 F77 编译器和 `-lf77compat` 链接选项创建的对象文件。

Fortran 的传统数组内部运行时库

自 2005 年的 Sun Studio 10 发行版开始，Studio Fortran 编译器就已不使用 SPARC 平台上的库 `libfmaxlai`、`libfmaxvai`、`libfminlai`、`libfminvai`、`libfprodai` 和 `libfsumai`。

未来发行版中可能会删除这些库。到那时，将不再能使用 Sun Studio 10 发行版之前的 Studio 编译器生成的对象文件和可执行文件，必须使用更新的 Studio 编译器重新编译。如果您有旧的对象文件和可执行文件需要其中的任何库并且无法重新编译，则应保留旧的编译器安装，或者将所需的特定库从旧编译器安装复制到新编译器安装。

`collect -R` 选项

`collect -R` 选项以前用于显示性能分析器工具新功能的 README 文件。此信息发布在《Oracle Solaris Studio 12.4 新增功能 》中的第 3 章 “性能分析工具”中。

第三方软件信息

Oracle Solaris Studio 12.4 软件包括受《Oracle Solaris Studio 12.4: Third Party Notices and Licenses 》制约的第三方技术。

发送使用数据到 Oracle

“发送使用数据到 Oracle”功能会定期将您的 Oracle Solaris Studio 组件使用信息发送给 Oracle Corporation。Oracle Corporation 使用此信息改进未来的 Oracle Solaris Studio 软件发行版。此信息是匿名的，不会与任何个人或组织关联。

但是，如果您希望禁用“发送使用数据到 Oracle”，请将 `SUNW_NO_UPDATE_NOTIFY` 环境变量设置为除零 (`0`) 之外的任何值。

此发行版的文档

以下文档随此 Oracle Solaris Studio 12.4 发行版提供。

- 《Oracle Solaris Studio 12.4：安装指南》。该文档包括通过 tar 文件、图形软件包安装程序和命令行软件包安装程序安装的信息。
- 《Oracle Solaris Studio 12.4 新增功能》。此文档介绍 Oracle Solaris Studio 12.4 发行版的新增功能。
- 《Oracle Solaris Studio 12.4：概述》。此文档提供有关如何使用编译器和工具以及它们如何配合使用的一般说明。
- 联机帮助。联机帮助可以通过 IDE 中的 "Help"（帮助）菜单来访问，提供关于使用 IDE 中所有组件的面向任务的信息。可通过每个工具的 "Help"（帮助）菜单获得性能分析器、线程分析器、代码分析器和 `dbxtool` 的联机帮助。
- 手册页。这些联机参考手册页介绍了用户命令、随编译器提供的库以及其他类型的命令。手册页包含参考信息，包括命令语法、用法以及相关命令。

  可通过在安装的 Oracle Solaris Studio 12.4 软件中使用 `man` 命令访问这些文档。
- Oracle Solaris Studio 12.4 手册和教程。可通过文档库（网址 `http://docs.oracle.com/cd/E37069_01/`）访问这些文档的 HTML 格式或 PDF 格式

此发行版中的已知问题、限制和解决方法

本节介绍此 Oracle Solaris Studio 12.4 发行版的部分已知问题，并提供有关如何解决这些问题的信息。

编译器问题

本节介绍了此发行版中编译器的已知问题、问题和解决方法。

编译器共有的问题

本节介绍适用于 `cc`、`CC` 和 `f95` 编译器的已知问题。

使用保留处理器的 OpenMP 处理器绑定问题

OpenMP 处理器绑定不遵守在 Linux 上使用 `taskset` 命令的处理器保留。

C++ 编译器问题

本节介绍了此发行版中 C++ 编译器的已知问题、问题和解决方法。

C++11 标准头

以下头适用于并发性功能，在 Oracle Solaris Studio 12.4 发行版中不受支持：

- `<atomic>`
- `<future>`
- `<thread>`
- `<mutex>`

编译 BOOST 库问题

不是所有 BOOST 库都可成功地进行编译。获得的结果可能与所用的 BOOST 版本以及库使用方式有关。BOOST 库包括在 Oracle Solaris Studio 编译器中；正在进行测试和改进以实现编译所有 BOOST。如果运行中遇到无法继续的问题，请报告该问题。

Oracle Solaris 上的 Apache 标准库问题

Oracle Solaris 10 8/11 及更高版本中以及在 Oracle Solaris 11、11.1 和 11.2 的初始发行版中安装的 Apache `stdcxx` 库无法正确使用新编译器缺省设置 `-template=no%extdef`。可能需要将选项 `-template=extdef` 添加到使用此库的 `CC` 命令行中。稍后的 Oracle Solaris 更新将提供针对此库问题的修复。有关更多信息，请参见《Understanding the Effects of the Changed Default C++ Template Compilation Model (`http://www.oracle.com/technetwork/articles/servers-storage-dev/changed-default-cpp-template-model-2292727.html`)》（《了解更改后的缺省 C++ 模板编译模式的效果》）。

Oracle Solaris 10 8/11 中安装的 Apache `stdcxx` 库在头 `stdcxx4/loc/_moneypunct.h` 中有语法错误。较早版本的编译器没有发现此错误，此错误是由 Oracle Solaris Studio 12.4 C++ 编译器造成的。无法禁用错误检测。

注 - 该语法错误问题已在 Oracle Solaris 10 1/13 和 Oracle Solaris 11.1 中修复。

多义性：构造函数调用或指向函数的指针

某些 C++ 语句有可能解释成声明或者表达式语句。C++ 消除歧义规则为：如果一个语句可以处理成声明，那么它就是声明。

在 Oracle Solaris Studio 12.2 及更早版本中，编译器会错误地解释如下代码：

```
struct S {
  S();
};
struct T {
```

```
  T( const S& );
};
T v( S() );    // ???
```

编程人员也许本来打算在最后一行定义变量 v，并且用类型为 S 的临时变量对它进行初始化。早期版本的编译器会这样解释这条语句。

但是在声明环境里，构造符号 "S()" 也可以是抽象声明符（不带标识符），表示“没有返回值类型为 S 的参数的函数”。在这种情况下，该语句会自动转换为函数指针 "S(*)()"。这样该语句仍可作为函数 v 的声明，该函数有一个函数指针类型的参数，返回类型为 T 的值。

当前版本的编译器可以正确地解释该语句，但这未必是编程人员所需要的结果。

可以使用两种方法来修改上述代码以便不产生歧义：

```
T v1( (S()) );  // v1 is an initialized object
T v2( S(*)() ); // v2 is a function
```

第一行中另加的圆括号表明它不是 v1 作为函数声明的有效语法，所以它的唯一可能解释是“利用类型为 S 的临时值进行初始化的类型为 T 的目标”。

同样，构造符号 "S(*)()" 不可能是一个值，所以它的唯一可能解释是函数声明。

第一行也可以改写为：

```
T v1 = S();
```

虽然这时语句含义非常清楚，但这种形式的初始化有时会创建一个额外的临时变量，而一般情况下不会发生这种情况。

建议不要编写与下面语句类似的代码，因为它的含义不清楚，不同的编译器可能会提供不同的结果。

```
T v( S() ); // not recommended
```

名称改编链接问题

以下条件可能会导致链接问题，并且仅在使用缺省 -compat=5 时适用。

- 函数在一个地方声明带有一个 const 参数，而在另一个地方又声明带有一个非 const 参数。

  示例：

```
void foo1(const int);
void foo1(int);
```

这两个声明是等效的，但编译器会将其改编为两个不同的名称。要避免这个问题，则不应将值参数声明为 `const`。例如，在任何位置都使用 `void foo1(int)`，包括该函数定义体。

- 函数有两个具有相同复合类型的参数，但只有一个参数是使用 `typedef` 声明的。

  示例：

```
class T;
typedef T x;
// foo2 has composite (that is, pointer or array)
// parameter types
void foo2(T*, T*);
void foo2(T*, x*);
void foo2(x*, T*);
void foo2(x*, x*);
```

  所有的 `foo2` 声明都是等效的，并且应该改编相同的名称。但是，编译器只会改编部分声明的名称。为了避免这个问题，应该统一使用 typedef。

  如果您无法统一使用 typedef，解决方法是：在定义该函数的文件中使用弱符号，使得声明与函数的定义一致。例如：

```
#pragma weak "__1_undefined_name" = "__1_defined_name"
```

  请注意，某些改编名称依赖于目标体系结构。（例如，在 SPARC V9 体系结构 (`-m64`) 中，`size_t` 是 `unsigned long`，而在其他体系结构中是不带符号的 `int`。）在这种情况下，会出现两个版本的改编名称，分别对应两个模式。这时必须使用两个 pragma，并用适当的 `#if` 指令对其进行控制。

  使用 `-compat=g` 或 `-std=c++11` 编译时不会出现本节所述情况。

不支持引用模板中的非全局名称空间目标

如果您使用 `-instances=extern` 编译，则使用模板和静态对象的程序会出现未定义符号的链接时错误。使用缺省设置 `-instances=global` 则不会出现问题。编译器不支持对模板中的非全局名称空间作用域目标的引用。请看以下示例：

```
static int k;
template<class T> class C {
        T foo(T t) { ... k ... }
};
```

在本示例中，一个模板类的成员引用了静态名称空间作用域的变量。请记住，名称空间作用域包含文件作用域。编译器不支持模板类的成员引用静态名称空间作用域的变量。另外，如果模板在其他的编译单元实例化，那么每个实例都会指向不同的 k，这破坏了 C++ 一次定义规则，代码的行为将会不可预测。

下面的方法也是可行的，但这取决于您如何使用 k，以及它应有的功能。第二个选项仅可供属于类成员的函数模板使用。

1. 可以为变量提供外部链接属性：

```
int k; // not static
```

   所有的实例都使用同一个 k。
2. 也可以使这个变量成为类的静态成员：

```
template<class T> class C {
        static int k;
        T foo(T t) { ... k ... }
};
```

   静态类成员具有外部链接属性。每个 C<T>::foo 的实例使用不同的 k。其他函数可以共享 C<T>::k 的实例。此选项可能是您需要的选项。

名称空间内的 #pragma align 需要改编名称

在名称空间内使用 #pragma align 时，必须使用改编名称。例如，在下面的代码中，#pragma align 语句是无效的。要更正此问题，应将 #pragma align 语句中的 a、b 和 c 替换为其改编名称。

```
namespace foo {
  #pragma align 8 (a, b, c) // has no effect
  //use mangled names: #pragma align 8 (__1cDfooBa_, __1cDfooBb_, __1cDfooBc_)
  static char a;
  static char b;
  static char c;
}
```

C 编译器问题

在此 cc 编译器发行版中应注意以下问题。

-xustr=ascii_utf16_ushort 选项与 -std=c11 不兼容

如果 -std=c11（编译器缺省选项）已生效，则指定标志 -xustr=ascii_utf16_ushort 会导致错误。

要避免错误，必须指定一个选项，将编译器接受的 C 语言变种更改为较早版本的 C 语言。建议使用的选项为 `-std=c99` 或 `-std=c89`。

如果使用以下更改 C 语言变种的任意选项，也会接受 `-xustr=ascii_utf16_ushort` 选项：`-ansi`、`-Xc`、`-Xa`、`-Xt` 或 `-Xs`。

Fortran 编译器问题

在此 `f95` 编译器发行版中应注意以下问题：

- 不换行打印行末尾之前的空格不会影响输出位置。

  输出语句格式末尾的 X 编辑描述符不会影响输出记录中后续字符的位置。如果输出语句中有 **`ADVANCE='NO'`** 并且有更多字符要通过后续输出语句传送到同一记录，则会导致差异。

  在很多情况下，可以通过添加空白字符串编辑描述符，而不是 *n* **x** 编辑描述符来解决此问题。它们并不完全相同，因为空白字符串编辑描述符实际上会将空白字符包含在记录中，而 *n* **x** 仅跳过后 *n* 个字符，缺省情况下通常导致跳过的位置中产生空白。
- 一行中有两个连续 & 号的有效代码会被拒绝。

  Fortran 标准禁止有一个 & 号的空续行。但是，仍可以创建同一行中有两个 & 号的空续行，这不在标准限制范围之内。编译器不会处理这种情况，而会显示一条错误消息。解决方法是删除该行，这只会影响程序可读性而并不添加任何语义。BOZ 常量有时会被截断。
- 在某些相对更复杂的情况下，如数组构造，BOZ 常量可能会被截成 4 个字节的缺省整数大小，即使它所应指定给的相应项是一个 8 个字节的整数实体。解决方法是在数组构造中使用正确类型和种类的常量，而不要使用 BOZ 常量。
- 更正新发行版中的舍入算法可能会导致列表控制的、名称列表控制的和格式化写入的 `ROUND='NEAREST'` 和 `ROUCH='COMPATIBLE'` 的输出与以前发行版中的输出不同。该舍入差异应该仅在最低有效位上。

Fortran 77 库已删除

需注意 Oracle Solaris Studio 12.2 发行版已删除了废弃的 FORTRAN 77 库。这意味着使用依赖于共享库 `libF77`、`libM77` 和 `libFposix` 的传统 Sun WorkShop `f77` 编译器编译的旧的可执行文件将不会运行。

数组内部函数使用全局寄存器

数组内部函数 `OT_PRODUCT` 和 `MATMUL` 针对相应 SPARC 平台体系结构进行了高度优化。因此，它们使用全局寄存器 `%g2`、`%g3` 和 `%g4` 作为临时寄存器。

对于区间运算，还会影响数组内部函数 ANY、ALL、COUNT、MAXVAL、MINVAL、SUM、PRODUCT。

如果调用了上述所列的数组内部函数，则用户代码不应该认为这些寄存器可用于暂时存储。当调用数组内部函数时，这些寄存器中的数据将被覆盖。

归档库中的 F95 模块不包括在可执行文件中

调试器 dbx 要求编译中使用的所有对象文件都包含在可执行文件中。通常，无需用户执行额外操作，程序即可满足此要求。但使用含有模块的归档文件时例外。如果程序使用了一个模块，但没有引用模块中的任何过程或变量，则产生的对象文件不会包含对模块中定义的符号的引用。只有对目标文件中定义的符号具有引用时，链接器才会链接归档文件中的对象文件。如果没有此类引用，则可执行文件中将不包括对象文件。dbx 将在尝试查找与所用模块关联的调试信息时发出警告。对于缺少调试信息的符号，则无法提供有关这些符号的信息。

使用 -u 链接程序选项可以解决这个问题。此选项使用一个符号作为其选项参数。它会将该符号添加到未定义的链接程序符号集中，这就需要解析此符号。与模块关联的链接程序符号通常是模块名称，其所有字母均为小写，后面跟有一条下划线。

例如，为了强制包含模块 MODULE_1 的对象文件被归档文件采用，请指定链接程序选项 -u module_1_。如果使用 f95 命令进行链接，请在命令行上使用 -Qoption ld -umodule_1_。

Linux 平台上的 **gethrtime(3F)**

启用系统节能功能时，没有可靠方式可以精确地获得 AMD 处理器上的时钟速率。因此，使用基于 gethrtime(3F) 的计时函数（Fortran 编译器的 Linux 版 Solaris gethrtime(3C) 函数）在 Linux 平台上获得高精度实际时间的方法只有在禁用了节能功能的 AMD 系统才会精确。要禁用节能功能，可能需要重新引导系统。

工具问题

本节列出了调试工具和性能分析工具的已知限制。

dbx 限制和不兼容情况

dbx 具有以下限制：

- 使用 libC.so.5 或者 libC.so.4 的旧副本可能会在 C++ 异常区域中引起 dbx 问题。可能会出现关于错误的 stab 和未处理的异常等警告消息。

解决方法：在所有系统上安装最新的 libC.so.5。

- Fortran 用户应该用 -stackvar 选项进行编译，以便充分利用运行时检查。
- 某些程序可能无法正常使用 -stackvar 选项。在这样的情况下，请尝试使用 -C 编译器选项，它将在不使用运行时检查的情况下启用数组下标检查。
- dbx 命令行解释器是旧版本的 Korn shell (ksh)，不支持代码集独立 (Code Set Independence, CSI)。当在 dbx 命令行上键入多字节字符时，会发生解释错误。
- Linux OS 上不能使用以下 dbx 功能：
  - 修复并继续
  - 多线程应用程序上的性能数据收集。
  - 在下列事件中设置断点：
    - fault
    - lastrites
    - lwp_exit
    - sysin
    - sysout
    - sync
  - 索引 DWARF（编译器选项 -xs=no）
- 在 Linux 平台上调试程序时可能会发生下面的问题：
  - 要调试 32 位程序，必须使用 -x exec32 选项启动 dbx。
  - 在调用 exec() 时，dbx 无法跟踪 Linux 平台上的派生进程，或更改为新程序
  - Korn shell 中的管道操作符仅限于 Linux 平台。任何需要访问目标进程的 dbx 命令不能作为管道的一部分使用。例如，下面的命令可能会导致 dbx 挂起：

    ```
    where | head -1
    ```

    解决方法：
    - 按 Ctrl-C 组合键以显示新的 dbx 提示符。
    - dbx 可缓存大量信息，对于上述示例，下列命令序列会起作用：

      ```
      where
      where | head –1
      ```
    - 将命令输出重定向到文件，然后显示文件内容。

      ```
      (dbx) > bag where
      (dbx) cat bag
      ```
- dbx 不支持 GNU C 和 C++ 编译器的下列功能：
  - VL 数组（gcc 4.1 及之前）
  - OpenMP
  - RTTI
  - 模板定义

  - 缺省参数
  - `using` 指令
  - `friend`
- Oracle Linux 6 上的 `dbx` 存在以下问题：
  - 系统库中使用的间接引用符号有时会导致 `dbx` 在引用处不是实际函数处设置断点。
  - 在调试使用 `gcc` 4.4.4 编译器编译的代码时，请注意 `dbx` 看不到使用 `-D` 调试器选项定义的宏。
- Oracle Solaris Studio 支持调试信息的 stabs 格式，但 Oracle 已宣布可能最终将停用该格式以支持 DWARF 格式。Oracle 不需要为新增功能或现有功能的增强实施 stabs 支持。

  Stabs 格式不支持以下调试功能。
  - 新增的 C++11 功能。
  - 优化后代码中的参数和局部变量。
  - 宏（当使用 `-g3` 选项编译代码时）。
  - 调试信息的子集（通常在使用 `-xdebuginfo={...}` 编译时）。
- 有关将 `dbx` 连接到进程时的数据收集问题的信息，请参见“`dbx attach` 分析 (`collect -P`)” [26]。

性能分析器和 `er_print` 实用程序限制

本节介绍性能分析器工具和 `er_print` 实用程序的已知问题。

- 共享对象的 "Library Visibility"（库可见性）功能有时无法与过滤正常配合使用。
- 性能分析器和 `er_print` 找不到 jar 文件中嵌入的共享对象。此外，在有些情况下，性能分析器和 `er_print` 找不到 Java 类文件或源文件。要解决这些问题，请使用 `addpath` 指向包含相应文件的目录。有关更详细的信息，请参见性能分析器 "Help"（帮助）中的 "Troubleshooting"（故障排除）部分。
- 在比较实验时，可能会看到几个问题：
  - 过滤器可能无法正常工作。
  - "Source"（源）和 "Disassembly"（反汇编）视图显示包含度量和独占度量，而不是如预期地仅显示包含度量。
  - "Dual Source"（双源）视图中的超链接有时会无效。
  - 将 "Comparison Style"（比较样式）设置为 "Delta"（增量）时，在 "Source"（源）和 "Disassembly"（反汇编）视图中不显示增量值。
- 可能会看到与具有可变时钟速率的 CPU 相关的以下问题：
  - 在采用可变时钟速率的系统上，基于周期的计数器的硬件计数器分析度量在处理器以低于最大速率运行时会低报，并且度量会转换为时间。

  - 如果比较或聚合包含基于周期的计数器的硬件计数器数据的多个实验，并且在具有不同时钟速率的计算机上记录实验，则数据视图中显示的计时度量中，只有第一个实验的计时度量是正确的。不过，"Overview"（概述）中显示的度量是准确的。
  - 在采用了以不同时钟频率运行的多个 CPU 的系统上，将根据其中一个 CPU 的时钟速率处理实验，这可能导致对其他 CPU 的计数过多或不足。
- 分析在 SPARC T5、M5 和 M6 系统上使用许多线程的 OpenMP 应用程序时，可能生成非常大而无法读取的实验。解决方法是在运行 `collect` 之前设置环境变量 `SP_COLLECTOR_NO_OMP`。请注意，在设置此变量时，无法在结果实验中看到 OpenMP 构造和度量，"User"（用户）、"Expert"（专家）和 "Machine"（计算机）视图全都看起来一个样。
- 性能分析器会放弃那些在分析过程中没有记录调用堆栈的实验。
- 如果 `/etc/hosts` 中没有主机名，则性能分析器将无法启动。解决方法是确保主机名位于 `/etc/hosts` 中。

在远程主机上分析应用程序的限制

尚不支持使用口令短语远程登录 。

`collect` 实用程序

本节介绍了性能分析器工具的 `collect` 实用程序和数据收集的已知问题。

- 通过 JDK 1.7.0_59 使用 JDK 1.7.0_40 时，由于 JDK 中存在错误，对混合了 Java 和 C++ 的应用程序的分析可能不完整。从 Java 到 C++ 的调用的堆栈未正确展开。
- Linux 上的堆跟踪不跟踪对 `calloc` 的调用。
- 在有些情况下，使用 `~system=1` 属性进行硬件计数器分析可能导致应用程序失败，或者导致 Oracle Solaris 操作系统崩溃或重置。
- 收集计数数据 (`collect -c`) 对使用 `-std=C++11` 编译的二进制文件无效。

`er_kernel` 实用程序

本节介绍 `er_kernel` 实用程序的已知问题。

- `er_kernel -t` 有时运行的时间会比请求的间隔多几秒。
- `er_kernel` 拒绝在运行于 Oracle VM 下的 Oracle Solaris 系统上运行，以避免会导致重新引导的 Oracle VM 错误。
- DTrace 堆栈展开有时会省略某个帧，通常出现在叶函数位于其尾声中时。

dbx collector 分析

本节介绍使用 dbx collector 命令时的应用程序分析问题。

- 在 Java 上使用 dbx collector 有限制。无法为 dbx 目标指定 Java 类文件。因此，必须指定 JVM 作为目标，并指定类文件作为 dbx run 命令的一个参数。例如：

```
> dbx /path-to-your-jdk/bin/java
(dbx) collector enable
(dbx) collector java on
(dbx) run [JVM-options] [Java-class-file-to-execute]
```

- 在 Linux 上，使用 dbx 或 collect -P 分析任何 Java 应用程序都可能失败。
- 在 dbx 中执行硬件计数器分析时，必须先发出命令 collector hwprofile on，然后再发出 collector hwprofile counter 或 collector hwprofile addcounter 命令以指定 hi 或 lo。否则，将报告错误。

  例如：

```
> dbx target-program
(dbx) collector hwprofile on
(dbx) collector hwprofile counter hi

> dbx target-program
(dbx) collector hwprofile on
(dbx) collector hwprofile addcounter lo
```

dbx attach 分析 (collect -P)

本节介绍使用 dbx attach 命令时的运行中应用程序分析问题：

- 如果将 dbx 连接到正在运行且启动时未使用 LD_PRELOAD 预先装入收集器库 libcollector.so 的进程，则将发生一系列错误。无法收集任何跟踪数据：同步等待跟踪、堆跟踪、I/O 跟踪或 MPI 跟踪。跟踪数据是通过对各个库执行插入操作而收集的。如果没有预先装入 libcollector.so，将无法执行插入操作。
- 如果在 dbx 连接到进程之前目标程序安装了信号处理程序，并且信号处理程序未在 Oracle Solaris 上传递 SIGPROF 和 SIGEMT 信号（或者未在 Linux 上传递 SIGIO 信号），则可能会丢失部分或全部分析数据。
- 如果目标程序使用 libcpc.so，则硬件计数器实验可能会失败，这是因为收集器和程序都在使用该库。
- 如果目标程序调用 setitimer(2)，则由于收集器和程序同时使用计时器，可能会使时钟分析实验中断。
- 连接到正在调用 malloc 库的目标可能导致目标失败。在连接时请求硬件计数器会极大地提高此类失败的概率。

- 除非使用 `collector java on` 命令或 `collect -j on -P` *pid*，否则无法连接到 Java 程序。
- 在 Linux 上，使用 `dbx` 或 `collect -P` 连接到 Java 程序可能会失败。
- 在 Linux 上，连接到正在进行阻塞调用或非阻塞调用的目标可能导致目标失败。
- 在 Linux 上，连接到多线程目标将不会正确记录在连接时已创建的线程的数据。不提供有关丢失数据的警告。请注意，其中包括所有 Java 目标，因为 JVM 是多线程的。

IDE 限制

本节介绍 IDE 中的已知限制。

要在远程主机上根据二进制文件创建项目，远程主机必须提供 Oracle Solaris Studio 12.4 或 Oracle Solaris Studio 12.3。较早的发行版不支持此功能。

代码分析工具限制

本节介绍代码分析器工具的已知限制。

- 当程序大小（文本 + 数据）大到超过 `-xmodel=small` 的最大范围 (2^{32} - 2^{24} - 1) 时，`discover` 和 `uncover` 工具无法处理使用 `-xmodel=medium` 编译的 x86_64 二进制文件。
- `discover` 工具对辅助文件无效；在 Oracle Solaris 11.2 中，辅助文件包含使用链接程序选项 `--z ancillary` 创建的调试信息。
- `discover` 工具不报告使用 C++11 编译的 UMR 和 PIR。
- 要在 Oracle Linux 上使用任何代码分析工具，都必须编译原始二进制文件并使用 `-xannotate=[yes]` 选项进行链接。