Oracle® Developer Studio 12.5：GCC 兼容性指南

2016 年 7 月

文件号码 E71995

本文介绍了 GCC（Gnu 编译器集合）与 Oracle Developer Studio 编译器和工具的兼容性。

本指南的目标读者是希望将通常用 GNU C 和 C++ 编译器编译的源代码用 Oracle Developer Studio C 和 C++ 编译器进行编译的用户。此策略可能包括分别将 Oracle Developer Studio 和 GNU 编译器用于同一应用程序的不同部分。如果用户希望保证源代码在两类编译器之间的兼容性，使用此策略也非常有用。

在本文中，GCC 一词是指 Gnu 编译器集合，其中包括 C、C++ 以及其他编译器。`gcc` 是指 GNU C 编译器，而 `g++` 是指 GNU C++ 编译器。

注 - 尽管本文中讨论了许多实现细节，但如需了解有关支持的功能的确切信息，还请参见《Oracle Developer Studio 12.5：C 用户指南》和《Oracle Developer Studio 12.5：C++ 用户指南》。

一般兼容性概念

本节介绍了影响 Oracle Developer Studio 和 GCC 编译器之间的兼容性的基本概念。

平台和 ABI

C 或 C++ 语言标准并未定义任何正式的 ABI（应用程序二进制接口）；但是，在特定平台上以及在使用特定指针大小时，可通过事实上的 ABI 在使用 C 级别导出接口编写的模块之间实现兼容性。

在此上下文中，平台是指以下两项的组合：

- 操作系统（例如 Oracle Linux 或 Oracle Solaris）
- 芯片系列（SPARC 或 x86）

指针大小是指二进制数据是使用 32 位 ABI 还是 64 位 ABI 构建的。某些平台可能仅支持一种指针大小，但目前，Oracle Developer Studio 产品支持的大多数平台都同时支持 32 位和 64 位程序。

Oracle Developer Studio 和 `gcc` 支持使用 `-m32` 和 `-m64` 选项来分别选择 32 位和 64 位 ABI。在 Solaris 10 和 Oracle Solaris 11 上，缺省模式为 32 位。在 Oracle Linux 上，缺省模式为 64 位。

要实现 C++ 语言的许多功能，编译器必须生成特定于编译器实现且多供应商平台范围 ABI 文档并未介绍的 ELF 符号以及其他二进制数据。

编译器的某些功能是由编译时或链接时选项选择的，这些功能可能会导致额外的外部符号引用，或导致编译器的二进制输出中发生超出 ABI 中定义的常规 C 级别接口的其他变化。对于这些功能，您可能需要将程序与用来创建对象文件的同一个编译器链接起来。某些功能还可能会阻止链接来自其他编译器的对象文件。

兼容性摘要

通常，您可以使用 Oracle Developer Studio 或 `gcc` 来编译 C 源文件，并自由地混合它们的对象文件来链接可执行文件或共享库。本节介绍了一些例外情况。

使用 C++ 时，您可以选择 Oracle Developer Studio C++ 编译器的 `g++` 兼容模式，并混合使用不同编译器编译的共享库和可执行文件，但您无法混合使用来自不同编译器的对象文件。下面将介绍详细信息。

Oracle Developer Studio 12.5 发行版存在一个涉及 `g++` 兼容性的具体问题，即 4.*x* 库和 5.*x* 库 ABI 之间存在兼容性问题。有关更多信息，请参见“GNU ABI 兼容性” [13]。

ABI 引用

Oracle Developer Studio 编译器生成的二进制代码如以下各个文档中所述：

- SPARC ABI: http://sparc.org/technical-documents
- 《SPARC Assembly Language Reference Manual》（《SPARC 汇编语言参考手册》）：第 6 章 "Writing Functions"http://docs.oracle.com/cd/E53394_01/html/E54833/index.html（编写函数）
- x86 ABI：http://www.x86-64.org/documentation/abi.pdf
- 《Oracle Solaris 64-bit Developer's Guide》（《Oracle Solaris 64 位开发人员指南》）
  - AMD64 ABI Features（AMD64 ABI 功能）：https://docs.oracle.com/cd/E53394_01/html/E61689/fcowb.html
  - SPARC V9 ABI Features（SPARC V9 ABI 功能）：https://docs.oracle.com/cd/E53394_01/html/E61689/advanced-2.html

汇编程序兼容性

SPARC 和 x86 汇编程序在计算机代码和伪操作方面存在很大不同。因此，GNU 汇编程序 (gas) 和 Oracle Developer Studio 与 Oracle Solaris 汇编程序之间的兼容性问题在不同平台上也是不同的。

用来生成 Oracle Solaris（SPARC 和 x86）附带的汇编程序和 Oracle Developer Studio（SPARC 和 x86）附带的汇编程序的源代码都是相同的，因此使用 Oracle Developer Studio 或 Oracle Solaris 汇编程序时，您预计会遇到类似的兼容性问题。

x86 汇编程序

在 Oracle Developer Studio 和 Oracle Solaris 汇编程序和 GCC 汇编程序之间切换时，请注意以下 x86 汇编程序问题。

- gcc 通常可以推断出操作代码后缀，而 Oracle Developer Studio 则要求显式提供操作代码后缀。提供更明确的操作代码后缀可同时满足两者的要求。例如，将 "mov" 更改为 "movw" 以及将 "shr" 更改为 "shrw"。
- '#' 可在 gcc 汇编程序文件中引入注释，但它以往也曾用来在 Oracle Developer Studio 汇编程序文件中引入预处理指令。

有关详细信息，请参见《x86 Assembly Language Reference Manual》 (https://docs.oracle.com/cd/E53394_01/html/E54851/index.html)（《x86 汇编语言参考手册》）

SPARC 汇编程序

官方 SPARC 汇编格式在《SPARC Assembly Reference Language Manual》 (http://docs.oracle.com/cd/E53394_01/html/E54833/index.html)（《SPARC 汇编参考语言手册》）中进行了定义。

与 ELF 部分有关的汇编语言指令

在 SPARC 汇编程序中，.section 指令使用的参数有所不同。与 GNU 汇编程序 (gas) 一样，属性标志是使用显式标记而不是使用字符串定义的。例如，使用 gas 时，.section 指令如下所示：

```
.section .init,"aw"
```

使用 Oracle Developer Studio 或 Oracle Solaris SPARC 汇编程序时，指令如下所示：

```
.section ".init",#alloc,#write
```

注 - SPARC 汇编程序支持 `.pushsection` 和 `.popsection` 指令，但不支持 `.previous` 指令。

伪操作问题

为确保 GNU 兼容性，SPARC 汇编程序中支持 `.symver` 伪操作。

`.uleb128` 和 `.sleb128` 伪操作都支持使用。

SPARC 汇编程序资源

有关详细信息，请参见以下资源：

- 当前 SPARC 指令集的说明：http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/sparc-processor-2516655.html
- SPARC ABI 文档：http://sparc.org/technical-documents/

头文件兼容性

GCC 和 Oracle Developer Studio 编译器预定义了不同的符号。

要查看 `gcc` 为 C 和 C++ 预定义的符号，请执行以下操作：

```
$ gcc -E -dM -xc /dev/null
$ g++ -E -dM -xc++ /dev/null
```

要查看 Oracle Developer Studio C 和 C++ 编译器预定义的符号，请执行以下操作：

```
$ cc -xdumpmacros -E /dev/null
$ CC -xdumpmacros -E /dev/null
```

使用各种编译器选项可能会影响预定义的宏及其在所有编译器中使用时的值。`-m32`|`-m64` 选项和语言选项 (`std=`*v*) 尤其会影响预定义的宏。

此输出也包括在 `_GNU_SOURCE` 宏中定义的源代码，该宏用来为各种不可移植的函数（大多在 `glibc` 中定义）启用头文件声明。有关更多信息，请参见 http://stackoverflow.com/questions/5582211/what-does-define-gnu-sourceimply。

预处理程序兼容性

Oracle Developer Studio C 和 C++ 编译器内置了自己的 C 预处理程序实现，并会在缺省情况下使用该预处理程序。这些预处理程序具有下列与 `gcc` 兼容的扩展。

Oracle Developer Studio 预处理程序

- `#warning`
- `#include_next`（用来实现包装头）

某些代码如果依赖于传统模式的预处理程序行为，则也会在 Oracle Developer Studio 和 GCC 编译器之间遇到差异。gcc 的传统模式如 https://gcc.gnu.org/onlinedocs/cpp/Traditional-Mode.html 所述。

在 Oracle Solaris 上，如果需要实现传统行为，您也可以使用 /usr/lib/cpp 加以实现，但最佳做法是将这些依赖关系替换为更现代的用法。/usr/lib/cpp 在设计时并不是一个打算与 gcc 完全兼容的预处理程序。

下文显示了有时会在源代码中看到的示例。

例 1 使用空注释的标记粘贴

```
FOO/**/BAR
```

此时输出中应该会出现 “FOOBAR”。但在 gcc 或 Oracle Developer Studio 中，这种情况不会出现，除非采取了特殊步骤启用了传统模式。在 Oracle Developer Studio C 中，这意味着指定 -Xs 选项。在 gcc 中，您必须指定 -traditional-cpp 选项。最好更新代码，使用 C 和 C++ 中定义的标准 ## 标记粘贴运算符。

例 2 标记之间的间距

```
#define FOO foo
#define BAR bar
FOO-BAR
```

这里预计会输出 “FOO-BAR” 而不是 “FOO - BAR”。预处理程序可能会（也可能不会）在 '-' 符号前后添加额外空格。gcc 编译器不会添加空格，但将编译器用作预处理程序并指定了 -E 或 -P 选项时，某些代码会依赖于该行为。要实现传统行为，您可以将 -Xs 与 Oracle Developer Studio C 编译器一起使用（但在 Oracle Developer Studio C++ 中不受支持），或者可以直接使用 /usr/lib/cpp。

编译器兼容性

本节介绍编译器功能以及影响 Oracle Developer Studio 和 GCC 之间兼容性的其他行为。

实现定义的行为

C 和 C++ 标准中的某些部分保留为“实现定义的行为”。这些详细信息在 GCC 和 Oracle Developer Studio 文档中定义，而且它们在某些方面存在一定差异。

位字段和枚举类型既可以带符号，也可以不带符号，具体由编译器选择。对于 enum，编译器如何选择因枚举值列表而异。Oracle Developer Studio C 和 C++ 的行为与 gcc 的行为不同。

带符号和不带符号 int 位字段

声明为 int（而不是 signed int 或 unsigned int）的位字段可以由编译器使用带符号或不带符号类型实现。提取值并决定是否对其进行符号扩展时，编译器如何选择便会体现出不同。

Oracle Developer Studio 编译器为 int 位字段使用不带符号类型，而 gcc 编译器使用带符号类型。使用 gcc -funsigned-bitfields 标志可控制此行为。

有关更多信息，请参见 https://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/Non_002dbugs.html 上的第六个列表项。

带符号和不带符号 enum 类型

C 和 C++ 中的 enum 类型可由编译器使用带符号或不带符号类型加以实现。它可能会依赖于特定类型的枚举值，因此很难预测。在这方面，Oracle Developer Studio 编译器的行为与 gcc 不同。Oracle Developer Studio 编译器为 enum 使用带符号值，而 gcc 编译器使用不带符号类型，除非 enum 类型的某些值明确分配了负值。

了解声明为 enum 类型的位字段的实现差异留给读者去做练习。最好更新源代码，使其具有更好的可移植性。对于位字段，您只需根据代码编写方式，将 “int” 声明更新为 “signed” 或 “unsigned” 即可。对于 enum 类型，您可以在需要将其转换为整数且不依赖于缺省类型时转换该类型。

C 语言扩展

Oracle Developer Studio 中的许多 C 语言扩展还通过 Oracle Developer Studio C++ 进行了实现。《Oracle Developer Studio 12.5：C 用户指南》 中的 “扩展”中介绍了一些 C 扩展。有关 gcc 语言扩展的列表，请参见 GCC 文档中的 Extensions to the C Language Family（C 语言系列的扩展）。

gcc 表中列出的部分项目现在是相关语言标准的标准功能，并在表 1 “在 Oracle Developer Studio 中实现的 GCC 扩展”中列为 C11、C99、C++03、C++11 和 C++14。

根据较早的语言标准编译时，较新的语言标准的功能通常作为扩展提供，除非它们与现有代码冲突，或者您使用选项启用了严格一致性模式。有关不同模式中可用的功能的更多信息，请参见该编译器的文档。

下表列出了在 Oracle Developer Studio 中实现的扩展。

表 1　在 Oracle Developer Studio 中实现的 GCC 扩展

GCC 扩展	Oracle Developer Studio 实现状态
语句表达式：将语句和声明放在表达式中。	已实现。
局部标签：块的局部标签。	仅在 C 中实现。
作为值的标签：获取标签指针及计算的 gotos。	仅在 C 中实现。
嵌套函数：就像在 Algol 和 Pascal 中一样，函数的词法作用域。	未实现。
构造调用：向另一个函数发出调用。	未实现。
Typeof：指的是表达式类型。	已实现。Oracle Developer Studio 12.5 C++ C++11 中的新功能，为此定义了 decltype。在 Sun (-compat=5) 模式下，C++ 会忽略 typeof 关键字，但仍支持 __typeof 和 __typeof__。
条件：忽略 ‘?:’ 表达式的中间操作数。	仅在 C 中实现。
__int128：128 位整数 – __int128。	未实现。
Long Long：双字整数 – long long int。	已在 C 和 C++ 中实现。
Complex：复数的数据类型。	非类型化 _Complex 缺省为 double，以便与 gcc 兼容。仅在 C 中实现。
浮点类型：其他浮点类型。	Oracle Developer Studio C 和 C++ 实现了 128 位 long double 类型，但没有实现名为 __float128 的类型。
半精度：半精度浮点。	未实现。

GCC 扩展	Oracle Developer Studio 实现状态
十进制浮点：十进制浮点类型。	未实现。
十六进制浮点：十六进制浮点常量。	C99。仅在 C 中实现。
定点：定点类型。	未实现。
命名地址空间：命名地址空间。	未实现。
零长度：零长度数组（常规零长度数组）。	`int foo[0];` 已在 C++ 的 GNU 兼容性模式下实现，或使用 `-features=zla` 实现。
零长度：零长度数组（弹性数组成员）。	结构结尾的 `int foo[]; //` 已在 C++ 的 GNU 兼容性模式下实现，或使用 `-features=zla` 实现。
空结构：不含成员的结构。	已实现。使用 C 时，要求 `-features=extensions`。
可变长度：长度在运行时计算的数组。	C99。已在 C 和 C++ 中实现。
可变参数宏：具有可变数目的参数的宏	C99。已在 C 和 C++ 中实现。Oracle Developer Studio 实现了 `gcc` 扩展，为可变宏参数提供用户定义的名称。缺少可变参数的 `gcc` 扩展未在 C++ 中实现。
转义换行符：转义换行符的规则稍有点宽松。	未实现。
下标：任何数组都可以带有下标，即便数组不是 `lvalue`。	C99。标准 C++。已在 C 和 C++ 中实现。
指针算术运算：对空指针和函数指针执行算术运算。	仅在 C 中实现。系统会生成警告。
数组指针：具有限定符的数组指针会按预期工作。	未实现。
初始化程序：非常量初始化程序。	C99。标准 C++。已实现。
复合文字：复合文字用来向结构、联合或数组提供值。	C99。已在 C 中实现。
指定的初始化程序：标记初始化程序的元素。	C99。已在 C 中实现。
case 范围：`` `case 1 ... 9` `` 等。	已实现。
转换为联合：从联合中的任何成员转换为 `union` 类型。	未实现。
混合声明：混合声明和代码。	C99。标准 C++。已实现。
属性扩展	`__has_attribute（）` 可用于测试已识别的属性。有关更多信息，请参见“元素组属性” [9]。
函数原型：原型声明和旧风格定义。	仅在 C 中实现。与 C++ 无关。
C++ 注释：可识别。	已在 C 中实现。
美元符号：标识符中允许使用美元符号。	已实现。要求 `-features=iddollar`。
字符转义：‘\e’ 代表字符 <ESC>。	未实现。
对齐：查询类型或变量对齐。	C11，拼写为 `_Alignof`。C 和 C++ 中支持 `__alignof__`。C++ 在 C++11 模式下支持 `alignof`。
内联：定义内联函数（速度与宏一样快）。	C99 和标准 C++。已实现。GCC 实现该标准前，它会创建一个静态函数体而不是外部函数体。如果您的代码依赖于此，您可以使用 C 选项 `-features=no%extinl` 从 Oracle Developer Studio C 编译器获得类似的行为。
volatile：何种访问构成对 volatile 对象的访问。	已实现。与 GCC 兼容。
将汇编语言与 C 一起使用：供 C 与汇编程序进行交互的指令和扩展。	已实现。C 和 C++ 实现了 `gcc`-compatible `asm（）` 语句，包括约束、`asm（）` 标签和显式寄存器变量。
替代关键字：用于头文件的 `__const__`、`__asm__` 等。	已实现。Oracle Developer Studio 12.5 C++ 添加了 `__asm` 和 `__volatile` 关键字拼写。

GCC 扩展	Oracle Developer Studio 实现状态
不完整的枚举：`enum foo;`，后面带有定义。	C++11。已在 C 和 C++ 中实现。必须为 C++11 模式。
函数名称：当前函数名称的可列显字符串。	Oracle Developer Studio compirs 支持 `__func____FUNCTION__` 和 `__PRETTY_FUNCTION__`。
返回地址：获取函数的返回地址或帧地址。	未实现。
向量扩展：通过内置函数使用向量指令。	请参见“SIMD 向量支持” [8]。
Offsetof：实现 `offsetof` 的特殊语法。	未实现。
`__sync` 内置函数：用于原子内存访问的传统内置函数。	已实现。使用 `-xatomic=studio`。请参见“原子” [14]。（`gcc` 弃用了这些函数，而是使用标准 `__atomic` 内置函数。）
`__atomic` 内置函数：带有内存模型的原子内置函数。	已实现。Oracle Developer Studio 12.5 中的新功能。使用 `-xatomic=studio`。请参见“原子” [14]。
整数溢出内置函数：执行算术检查和算术溢出检查的内置函数。	未实现。
用于事务处理内存的 x86 专用内存模型扩展：x86 内存模型。	未实现。
对象大小检查：用于受限缓冲区溢出检查的内置函数。	未实现。
指针边界检查程序内置函数：用于指针边界检查程序的内置函数。	未实现。
Cilk Plus 内置函数：用于 Cilk Plus 语言扩展的内置函数。	未实现。
其他内置函数：其他内置函数。	`__builtin_constant_p（）` 可以测试某个项目是否为编译时常量。Oracle Developer Studio 尚未实现大多数其他内置函数。
目标内置函数：专用于特定目标的内置函数。	未实现。
目标格式检查：专用于特定目标的格式检查。	未实现。
Pragmas：`gcc` 接受的 Pragmas。	已实现 `#pragma once`。其他尚未实现。
未命名字段：结构/联合中的未命名结构/联合字段。	C++11。已在 C 和 C++ 中实现。
线程局部：每个线程的变量。	C99。C++11。已在 C 和 C++ 中实现。
二进制常量：使用 ‘`0b`’ 前缀的二进制常量。	仅在 C++ 中实现。

SIMD 向量支持

Oracle Developer Studio 支持某些特定于体系结构的内部函数通过使用特定于 CPU 的指令对向量操作进行编码。创建这些向量的数据类型时会使用 `vector_size` 属性或按“头文件兼容性” [4]中所述加入相应的头文件。部分 C 运算符支持这些类型，但在某些情况下，您可能需要使用特定于 CPU 的内部函数。

有关 `_m128` 等类型以及对这些类型的操作的说明，请参见《Oracle Developer Studio 12.5：C 用户指南》 中的 “SIMD 内部函数”中的“SPARC64 X”一节。

C++ 专用功能

Oracle Developer Studio C++ 编译器支持 `-features=cplusplus_redef` 选项，该选项允许 `__cplusplus` 宏使用使用非标准值，以便用于需要该设置的源代码。《Oracle Developer Studio 12.5：C++ 用户指南》 中的 “–xrestrict[=f]”

C++ 的部分其他 GNU 扩展包括：

- 枚举中的 `long long` 常量
- 允许为 `void` 使用 `typedef` 作为函数参数
- ```
  typedef void VOID;
  int foo(VOID);
  ```
- 包括 `<stdbool.h>` 时，`_Bool` 等同于 Oracle Linux 上的 `bool`
- `extern` 模板
- `long long` 位字段
- `pragma` pack push/pop

有关 C++ 扩展的信息，请参见 GCC 文档中的 C++ 语言的扩展。Oracle Developer Studio 编译器中相同功能的信息可在下表中找到。

表 2　　GNU C++ 扩展

GNU C++ 扩展	Oracle Developer Studio 实现状态
C++ Volatile：确定何种访问构成对 volatile 对象的访问。	兼容的实现。
受限指针：C99 受限指针和引用。	已实现。
Vague 链接：G++ 放置内联、vtable 等内容的位置。	Oracle Developer Studio 也为这些使用 COMDAT，但方法可能稍有不同。
C++ 接口：您可以同时为声明和定义使用单个 C++ 头文件。	未实现。在 `gcc` 中已弃用。
模板实例化：确保发出每个所需模板实例化的一份精确副本的方法。	Oracle Developer Studio 也为这些使用 COMDAT，但方法可能稍有不同。
绑定的成员函数：您可以提取一个指向由 '->*' 或 '.*' 表达式表示的方法的函数指针。	未实现。
C++ 属性：仅限 C++ 的变量、函数和类型属性。	请参见"元素组属性" [9]。
函数多版本：声明多个函数版本。	未实现。
命名空间关联：使用命名空间关联指令的强关联。	未实现。在 `gcc` 中已弃用，取而代之的是标准 C++11 功能。
traits 类型：编译器支持 traits 类型。	未实现。
C++ 概念：改善了对通用编程的支持。	未实现。
Java 异常：调整异常处理，使其能与 Java 一起使用。	未实现。
已弃用的功能：计划从 C++ 中移除。	未实现。
向后兼容性：兼容早期的 C++ 定义。	未实现。

元素组属性

有关 Oracle Developer Studio 12.5 编译器中实现的属性的完整说明，请参见《Oracle Developer Studio 12.5：C 用户指南》 中的 "支持的属性"和《Oracle Developer Studio 12.5：C++ 用户指南》 中的 "支持的属性"。

`__has_attribute（）` 内置函数可与 `gcc` 和 Oracle Developer Studio 编译器一起使用，从而测试已识别的属性。

下列属性是 Oracle Developer Studio 12.5 C++ 中新增的属性：`aligned`、`deprecated`、`weak(alias)`、`weakref`、`packed`、`tls_model`、`vector_size` 和 `visibility`。
  ```

Oracle Developer Studio 编译器不支持属性的所有语法。

表 3 GCC 属性

GCC 属性类别	Oracle Developer Studio 中支持的属性
函数属性：声明函数没有副作用，或它们从不返回任何内容。	`alias`、`aligned`、`always_inline`、 `const` `constructor/destructor`、 `deprecated` (C++14)、`malloc`、`noinline`、`noreturn`、 `nothrow`（仅限 C++）、`pure`、`returns_twice`、`visibility`、 `weak`（和 `alias`）、`weakref` (C++)
变量属性：指定变量属性。	`aligned`、`deprecated`、`mode` (C++)、`packed`（部分实现）、`tls_model`、`vector_size`、`weak`
类型属性：指定类型属性。	`aligned`, `deprecated`, `packed`, `visibility`
枚举器属性：指定枚举器属性。	`deprecated`

命令行选项

Oracle Developer Studio 和 `gcc` 编译器都实现了传统的编译器选项，例如 `-g`、`-c`、`-o` 等。下表介绍了 Oracle Developer Studio 编译器专为与 `gcc` 兼容而实现的选项。

Oracle Developer Studio 中的以下选项与 `gcc` 兼容。

表 4 兼容选项

选项	说明
`-std`	`-std` 选项会选择语言标准，例如 C11、C++11 等。
`-pedantic`	违反本应接受的技术标准时发出错误或警告。此选项是 Oracle Developer Studio 12.5 C++ 编译器中的新增选项。
`-m32` / `-m64`	选择 32 位或 64 位二进制输出。
`-shared`	生成共享库。在 Oracle Solaris Studio 12.4 中，`-shared` 选项所起的作用类似于 `-G` 的别名。在 Oracle Developer Studio 12.5 中，此选项所起的作用类似于 `gcc` 中的 `-shared` 选项。请参见表 5 “有差异的选项”。

以下选项在 Oracle Developer Studio 和 GCC 中的工作方式有所不同。

表 5 有差异的选项

Oracle Developer Studio Option	GCC 选项	说明
`-xM`	`-M`	列显源文件的 make 风格依赖关系。在 `gcc` 中，`-xM` 会执行另一个函数。在 Oracle Developer Studio 中，`-M` 会选择链接程序映射文件。
`-B`(static| dynamic)	`-Wl,-B`(static| dynamic)	GNU 链接程序支持此行为，但 `gcc` 不支持。使用 `-Wl` 将选项传递给 GNU `ld`。在 `gcc` 中，`-B` 会执行另一个函数。
`-G`	`-shared`	生成共享库时，`gcc` 中的 `-shared` 会添加 C++ 库依赖关系。Oracle Developer Studio 编译器中的 `-G` 选项不会添加它们。

以下 `gcc` 风格的选项会自动转换为 Oracle Developer Studio 中的对应选项。要查看选项列表，请将此选项传递给 C 或 C++ 编译器的以下选项：`-xhelp=gccflags`。

表 6　　自动转换的选项

Oracle Developer Studio 中的 `gcc` 选项	Oracle Developer Studio 编译器中的行为
`-MM`	与 `-xM1` 相同。
`-Ofast`	与 `-fast` 相同。
`-Wall`	与 +w2 相同（仅限 C++）
`-Wall`	与 `-v` 相同（仅限 C）
`-Werror`	与 `-errwarn=%all` 相同。
`-Wpedantic`	与 `-pedantic` 相同。
`-fno-elimininate-unused-debug-types`	接受并忽略
`-fopenmp`	与 `-xopenmp` 相同。
`-fPIC`	与 `-KPIC` 相同。
`-fpic`	与 `-Kpic` 相同。
`-fplan9-extensions`	接受并忽略（仅限 C）。
`-fplugin-arg-name=`*t*	警告并忽略（仅限 C）
`-fplugin=`*t*	警告并忽略（仅限 C）
`-fsigned-char`	与 `-xchar=signed` 相同（仅限 C）
`-fsyntax-only`	与 `-xe` 相同。
`-funsigned-char`	与 `-xchar=unsigned` 相同（仅限 C）
`-gdwarf-version`	与 `-xdebugformat=dwarf` 相同。
`-gstabs`	与 `-xdebugformat=stabs` 相同。
`-gstabs+`	建议使用 `-xdebugformat=stabs`；选项被忽略
`-iplugindir=`*t*	警告并忽略
`-march=`*a*	与 `-xtarget=` 相同 *a*
`-mcpu=`*a*	与 `-xtarget=` 相同 *a*
`-mfmaf`	与 `-fma=fused` 相同。
`-mno-vis`	与 `-xvis=no` 相同。
`-mno-vis2`	与 `-xvis=no` 相同。
`-mno-vis3`	与 `-xvis=no` 相同。
`-mtune=`*a*	与 `-xtarget=` 相同 *a*
`-mvis`	与 `-xvis` 相同。
`-mvis2`	与 `-xvis` 相同。
`-mvis3`	与 `-xvis` 相同。
`-no-canonical-prefixes`	接受并忽略
`-no-fma`	与 `-fma=none` 相同。
`-no-fmaf`	与 `-fma=none` 相同。
`-no-trigraphs`	与 `-xtrigraphs=no` 相同。
`-nodefaultlibs`	与 `-xnolib` 相同（仅限 C++）
`-pass-exit-codes`	警告并忽略
`-pedantic-errors`	与 `-pedantic` 相同。
`-pipe`	警告并忽略
`-save-temps`	与 `-keeptmp` 相同。
`-shared`	与 `-G` 相同。
`-specs=`*t*	警告并忽略
`-traditional`	与 `-Xs` 相同（仅限 C）
`-trigraphs`	与 `-xtrigraphs=yes` 相同。

对象文件兼容性

本节介绍影响 Oracle Developer Studio 和 GCC 之间兼容性的对象文件功能。

注释

部分 Oracle Developer Studio 工具依赖于编译器发出的有关已生成代码的其他结构信息。此信息称为“注解”，而注解会发送至名为 ".annotations" 的 ELF 部分。代码分析器的某些功能依赖于注解。生成此数据时可使用 -xannotate 选项进行控制。使用此信息的命令包括 binopt、code-analyzer、discover、collect 和 uncover。

C++ 改编名称

C++ 编译器会生成在其中编码了类型信息的 ELF 符号。这些符号称为“改编名称”。改编名称的格式属于实现细节，因此在 compat=5 (Sun ABI) 模式下编译的 Oracle Developer Studio 代码不会与使用 g++ 编译的代码正确混合。在 gnu 模式下使用 Oracle Developer Studio C++ 编译器会使得对象文件兼容。有关更多信息，请参见《Oracle Developer Studio 12.5：C++ 用户指南》。

使用检测的功能

需要检测对象代码的编译器功能也要求在链接时包含额外的库或对象文件。使用这些功能时，您需要将可执行文件或库与 Oracle Developer Studio 编译器链接起来。请注意，将 gcc 编译的代码混合到程序或库中时不会进行检测，因此结果将不完整。

此类别中的功能包括：

- 分析反馈 – 包括选项 -xprofile 和 -xlinkopt。
- 传统分析 – 包括编译时和链接时选项 -xpg。
- 传统覆盖 – 包括选项 -xprofile=tcov 和 tcov 实用程序。

调试信息

dbx 和其他 Oracle Developer Studio 工具会使用编译器生成的调试信息。这些信息大多以 DWARF 格式记录，但一些附加索引信息以名为 STABS 的旧格式记录。dwarf 的 ELF 部分以 ".debug" 开始，而 stabs 的 ELF 部分以 ".stab" 开始。

运行时支持兼容性

本节介绍与运行时支持有关的问题。

libgcc 支持

编译器在需要时会自动与 libgcc 链接，但如果出现问题，了解库的设计用途将很有帮助。gcc 支持库 (libgcc) 含有各种帮助程序例程，这些例程有两种格式，一种是称为 libgcc.a 的归档库，另一种是称为 libgcc_s.so 的共享库。libgcc 包含支持异常的例程以及支持整数和浮点运算的例程，这

些运算在当前体系结构中可能没有直接指令。该库的功能如 gcc 文档中的 GCC 底层运行时库中所述。

使用 gcc 编译器创建的可执行文件会与 libgcc.a 链接，而不会与 libgcc_s.so 建立动态依赖关系。使用 g++ 编译器创建的可执行文件会使用 libgcc_s.so，而且会与该库建立动态依赖关系。

Oracle Developer Studio C 编译器生成的代码不会与 libgcc 建立依赖关系。

在 Sun 模式 (-compat=5) 下编译时，Oracle Developer Studio C++ 编译器生成的代码不会依赖于 libgcc，但在 GNU 兼容性模式（-compat=g、-std=c++03、-std=c++11 或 -std=c++14）下编译时会具有此依赖关系。

在 Oracle Linux 上，名为 libc_supp.a 的类似库会随 Oracle Developer Studio 附带。

C++ 运行时支持

使用 Oracle Developer Studio C++ 编译器编译的源文件需要使用 Oracle Developer Studio C++ 编译器进行链接，以确保使用正确的运行时库、CRT 文件和链接程序选项。g++ 对象文件应与 g++ 链接。由于对象文件中使用了大量不兼容的实现细节，因此不支持将来自 Oracle Developer Studio C++ 和 g++ 编译器的对象文件混合到同一个可执行文件或共享库中。

GNU ABI 兼容性

Oracle Developer Studio C++ 在 GNU 兼容性模式（使用选项 -compat=g、-std=c++03、-std=c++11 或 -std=c++14）下创建的共享库可与 g++ 编译器创建的共享库混合，并链接至任一编译器创建的主程序，但它们必须使用相同版本的 g++ ABI。

在 GNU 兼容性模式下操作时，Oracle Developer Studio C++ 编译器与标准库对象的库接口、名称改编和二进制布局区域中的 g++ 代码具备二进制兼容性。但是，它仍为许多底层实现方面使用不同的机制。外部内联函数、模板实例、RTTI 记录和异常范围信息都使用 COMDAT 按不同方式实现。异常范围信息以 Studio 专用方式设置格式。这限制了使用 Oracle Developer Studio C++ 编译的对象文件和归档库与使用 g++ 编译的对象文件和归档库之间的混合程度。

GNU C++ ABI 5.*x* 变更

由于 4.*x* 和 5.*x* 存在一个不兼容的 GNU C++ ABI 变更，使用 g++ 编译的可执行文件和库必须统一使用一个 ABI 才能正常工作。5.*x* GNU C++ 库同时支持两个 ABI，但编译器（GCC 或 Oracle Developer Studio）一次只能生成一个 ABI。g++ 编译器缺省使用 5.*x* ABI.要选择 4.*x* ABI，编译时请使用 -D_GLIBCXX_USE_CXX11_ABI=0。

Oracle Developer Studio 12.5 C++ 编译器仅实现了 4.*x* ABI.要混合共享库和可执行文件，请使用上述选项编译 g++ 部分。有关此问题的 g++ 文档可在 https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html 找到。

从技术上来说，将 4.*x* 库或可执行文件与 5.*x* 库或可执行文件一起使用不存在任何障碍，但它们无法使用在两个 ABI 中不同的库类进行彼此调用。这些类中有一些很常用（例如字符串类），因此混合两个 ABI 起不到太大作用。

`main` 函数

C++ 中的 main 函数采用了某些额外的机制。编译对象文件及 main 时使用的编译器也应用来链接可执行文件。编译器驱动程序使用的 crt 对象文件与对象文件 main 中嵌入的机制匹配。

libCrunG3.so 库

在 GNU 模式下编译代码时，Oracle Developer Studio C++ 编译器会使用 GNU C++ 库，但它也需要链接名为 `libCrunG3.so` 的另一个库，以便为 RTTI、异常和其他语言功能提供支持函数。此库对应于在 Sun 模式 (`-compat=5`) 下使用的 `libCrun.so` 库。

不同版本的 GNU C++ 库

Oracle Developer Studio 12.5 C++ 编译器（在 GNU 兼容性模式下）使用 5.1.*x* 版本的 GNU C++ 库。您可以将使用不同版本的 GNU C++ 库编译的共享库和应用程序混合起来，但用于编译和链接主应用程序的编译器的版本至少要与用来创建任意共享库的编译器版本一样新。如果使用 Oracle Developer Studio C++ 编译了任何共享库，主应用程序应使用 Oracle Developer Studio C++ 或使用 `g++` 编译器（都支持 5.1.*x* GNU C++ 库）编译。

运行时库位置

C++ 运行时库位于各种位置。

以下 Sun 模式 (`-compat=5`) C++ 库位于 Oracle Solaris 上的 `/usr/lib`，但在 Oracle Linux 上并不位于该位置：

- `libCstd.so`
- `libCrun.so`
- `libdemangle.so`
- `libiostream.so`
- `libstdcxx.so`

以下 GNU 模式 C++ 库在 Oracle Solaris 上位于 `/usr/lib`，但在 Oracle Linux 上并不位于该位置：

- `libCrunG3.so`（但是，您可能需要修补 Oracle Solaris 软件包 `SUNWlibC`）
- `libdemangle.so`

所有其他 C++ 运行时库都位于编译器安装目录。

Oracle Developer Studio C++ 编译器针对 Oracle Developer Studio 捆绑的任意共享库进行链接时，它通常会将必要的 `RPATH` 设置添加到可执行文件中，以便在运行时查找库。您可通过添加 `-xnorunpath` 选项并添加相应的 `-R` 选项来防止此行为并配置 `RPATH` 设置。

OpenMP

使用 `-xopenmp` 编译源代码时会在系统库上生成运行时依赖关系，对 Oracle Developer Studio 编译器来说，该系统库就是 `libmtsk` 库。在 Oracle Solaris 上，此库位于 `/usr/lib` 中。在 Oracle Linux 上，此库位于编译器安装目录中。由于线程管理是一个进程范围内的操作，在一个程序中，请仅使用一个 OpenMP 实现。如果共享库使用 OpenMP，主应用程序将无法使用不同的 OpenMP 实现。Oracle Developer Studio 的 OpenMP 实现与 GCC 的 OpenMP 实现不兼容。

原子

原子是 C 2011 和 C++ 2011 标准中新增的语言功能，而且它需要来自操作系统的运行时支持。Oracle Developer Studio 编译器使用初步绑定的运行时库（该库与部分版本的 GCC 运行时库

兼容）支持此功能（请参见《Oracle Developer Studio 12.5：C++ 用户指南》 中的 “捆绑的原子库”），而且您可以使用 `-xatomic` 选项选择链接程序和库时使用哪个运行时库。

线程本地

GCC 和 Oracle Developer Studio 编译器实现了线程局部数据 (`__thread declarations`)，且采用的实现方法符合 OS ABI（Oracle Solaris 或 Oracle Linux），因此相关功能在 Oracle Developer Studio 和 GCC 编译器编译的代码之间兼容。有关适用于线程局部数据的 Oracle Solaris ABI 的更多信息，请参见《Oracle Solaris 11.3 链接程序和库指南》。

共享库兼容性

使用 Oracle Developer Studio C++ 编译器编译的源文件需要使用 C++ 编译器进行链接，以确保使用正确的运行时库和链接程序选项。G++ 对象文件不应使用 `g++` 编译器进行链接。

使用 Oracle Developer Studio CC 及选项 `-compat=g`、`-std=c++03`、`-std=c++11` 或 `-std=c++14` 创建的共享库可与使用 `g++` 编译器创建的共享库自由混合，并链接至由任一编译器创建的主程序。针对新版本 `gcc` 库编译的二进制在大多数情况下无法链接至旧版本 `gcc` 库。

Oracle Developer Studio 12.5 C++ 编译器在 `gcc` 兼容模式下使用 5.1.*x* 版本的 `g++` 运行时库。

您可以将使用不同版本的 `g++` 库编译的库和应用程序混合起来，但用于编译和链接主应用程序的编译器的版本至少要与用来创建任意共享库的编译器版本一样新。如果使用 Oracle Developer Studio C++ 编译了任何共享库，主应用程序应使用 Oracle Developer Studio C++ 或使用 `g++` 编译器（都支持 5.1.*x* 运行时库）进行编译。

工具兼容性

本节介绍了 Oracle Developer Studio 工具与 GCC 的兼容性。

dbx 调试器

`dbx` 可以调试使用 Oracle Developer Studio 和 GCC 编译器生成的 C 和 C++ 代码。它与“GCC 版本兼容性” [16] 中列出的 GCC 版本兼容。

性能分析器

性能分析器支持混合 Oracle Developer Studio 和 GCC 二进制。它可以分析 Oracle Developer Studio 和 GCC 编译器编译的代码，并支持“GCC 版本兼容性” [16] 中列出的 GCC 版本。

代码分析器

Discover ADI (`-i` *adi*) 和 Discover Lite (`-l`) 支持 Oracle Developer Studio 和 `gcc` 二进制。`discover`(1) 实用程序会在 Oracle Developer Studio 编译的代码中检测错误，但不会检查 `gcc` 编译的代码。`uncover`(1) 实用程序会收集 Oracle Developer Studio 编译的代码的覆盖信息。

IDE

IDE 项目支持只使用一个编译器（GCC 或 Oracle Developer Studio）编写或编译的代码。此限制的解决办法是为 GCC 编写的代码创建单独的项目。

GCC 版本兼容性

以下操作系统支持 GCC 4.8.2、4.9.2 和 5.1.0 版本：

- Solaris 10 1/13 (Update 11)
- Oracle Solaris 11.2
- Oracle Solaris 11.3
- Oracle Linux 6.6
- Oracle Linux 7.*x*

Oracle Linux 6.6 支持 GCC 4.4 版本。

更多参考信息

- Extensions to the C Language Family (`https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html`)（C 语言系列的扩展）
- Extensions to the C++ Language (`https://gcc.gnu.org/onlinedocs/gcc/C_002b_002b-Extensions.html#C_002b_002b-Extensions`)（C++ 语言的扩展）
- GCC Binary Compatibility (`https://gcc.gnu.org/onlinedocs/gcc/Compatibility.html`)（GCC 二进制兼容性）
- GNU C++ ABI Policy and Guidelines (`https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html`)（GNU C++ ABI 政策和准测）
- Intel® Compilers for Linux: Compatibility with GNU Compilers (`https://software.intel.com/en-us/articles/intel-compilers-for-linux-compatibility-with-gnu-compilers`)（适用于 Linux 的 Intel® 编译器：与 GNU 编译器的兼容性）

Oracle Developer Studio 12.5：GCC 兼容性指南

文件号码 E71995

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。

Part No: E71995