Oracle® Developer Studio 12.5: GCC 互換性ガイド

2016 年 7 月

Part No: E71994

このドキュメントでは、GCC (Gnu コンパイラコレクション) の Oracle Developer Studio コンパイラおよびツールとの互換性について説明します。

このガイドは、通常は GNU C および C++ コンパイラでビルドするソースコードを使用し、Oracle Developer Studio C および C++ コンパイラでビルドするユーザーのためのものです。この方法は、同じアプリケーションの異なる部分について、Oracle Developer Studio コンパイラと GNU コンパイラを使用することも含みます。さらに、この方法は、どちらの種類のコンパイラにも対応するようなやり方でソースコードを維持する必要があるユーザーにとっても有用です。

このドキュメントでは、GCC という用語は Gnu コンパイラコレクションを指すのに使用され、(特に) C および C++ コンパイラを含みます。`gcc` という用語は GNU C コンパイラを指し、`g++` は、GNU C++ コンパイラを指します。

注記 - 多くの実装の詳細はこのドキュメントで説明されていますが、サポートされている機能の最終的なソースは、『Oracle Developer Studio 12.5: C ユーザーズガイド』および『Oracle Developer Studio 12.5: C++ ユーザーズガイド』です。

互換性の一般的な概念

このセクションでは、Oracle Developer Studio コンパイラと GCC コンパイラの間の互換性に影響を与える基本概念について説明します。

プラットフォームおよび ABI

C および C++ の言語規格では、正式な ABI (アプリケーションバイナリインタフェース) が定義されていませんが、特定のプラットフォームおよびポインタサイズ内においては、事実上の ABI によって、C レベルのエクスポート対象インタフェースで記述されるモジュール間の互換性が可能になります。

このような文脈では、プラットフォームは次の 2 つの組み合わせとなります。

- オペレーティングシステム (たとえば、Oracle Linux、または Oracle Solaris)
- チップファミリ (SPARC または x86)

ポインタサイズは、バイナリが 32 ビット ABI または 64 ビット ABI のどちらを使用してビルドされているかを指します。一部のプラットフォームは 1 つのポインタサイズしかサポートしないこともありますが、Oracle Developer Studio 製品によって現在サポートされているほとんどのプラットフォームは 32 ビットプログラムと 64 ビットプログラムの両方をサポートします。

Oracle Developer Studio および `gcc` は、それぞれ 32 ビットおよび 64 ビットの ABI を選択するための `-m32` および `-m64` オプションをサポートしています。Solaris 10 および Oracle Solaris 11 では、デフォルトモードは 32 ビットです。Oracle Linux では、デフォルトモードは 64 ビットです。

C++ 言語の多くの機能を実装するには、コンパイラは、そのコンパイラ実装に固有で複数ベンダープラットフォーム全体に及ぶ ABI ドキュメントで扱われていない、ELF 記号およびその他のバイナリデータを生成する必要があります。

コンパイル時またはリンク時のオプションによって選択されたコンパイラの一部の機能は、ABI で定義される通常の C レベルインタフェースを超える、追加の外部シンボル参照またはコンパイラのバイナリ出力のその他の変更をもたらすことがあります。これらの機能については、オブジェクトファイルの作成に使用されるものと同じコンパイラを使用してプログラムをリンクすることが必要な場合があります。一部の機能によって、ほかのコンパイラからのオブジェクトファイルとのリンクが妨げられることもあります。

互換性に関するサマリー

一般に、Oracle Developer Studio または `gcc` で C 言語のソースファイルをコンパイルし、実行可能ファイルまたは共有ライブラリをリンクするために、それらのオブジェクトファイルを自由に混在させることができます。このセクションでは、いくつかの例外について説明します。

C++ の場合、Oracle Developer Studio C++ コンパイラの `g++` 互換モードを選択し、ほかのコンパイラでビルドされた共有ライブラリおよび実行可能ファイルを混在させることができますが、ほかのコンパイラによるオブジェクトファイルを混在させることはできません。詳細については後述します。

Oracle Developer Studio 12.5 リリースには、4.*x* ライブラリと 5.*x* ライブラリの ABI の間で `g++` の互換性に関する特定の問題があります。詳細については、14 ページの「GNU ABI の互換性」を参照してください。

ABI の参考資料

Oracle Developer Studio コンパイラによって生成されたバイナリコードは、さまざまなドキュメントで説明されています。

- SPARC ABI: `http://sparc.org/technical-documents`
- SPARC アセンブリ言語リファレンスマニュアル: 第 6 章、書き込み関数 `http://docs.oracle.com/cd/E53394_01/html/E54833/index.html`
- x86 ABI: `http://www.x86-64.org/documentation/abi.pdf`
- Oracle Solaris 64 ビット 開発者ガイド
  - AMD64 ABI 機能: `https://docs.oracle.com/cd/E53394_01/html/E61689/fcowb.html`
  - SPARC V9 ABI 機能: `https://docs.oracle.com/cd/E53394_01/html/E61689/advanced-2.html`

アセンブラの互換性

SPARC アセンブラと x86 アセンブラは、マシンコードの観点や擬似オペランドの観点で動作が大きく異なっています。したがって、GNU アセンブラ (`gas`) と、Oracle Developer Studio アセンブラおよび Oracle Solaris アセンブラとの間の互換性の問題は、異なるプラットフォームでは違った問題になります。

Oracle Solaris (SPARC および x86) とともに出荷されるアセンブラと、Oracle Developer Studio (SPARC および x86) とともに出荷されるアセンブラを生成するのに同じソースコードが使用されるため、Oracle Developer Studio アセンブラまたは Oracle Solaris アセンブラのいずれかを処理するときは、類似した互換性の問題が想定されます。

x86 アセンブラ

Oracle Developer Studio アセンブラおよび Oracle Solaris アセンブラと、GCC アセンブラとの間で切り替えるときに、x86 アセンブラに関する次の問題に注意してください。

- `gcc` は命令コードの接尾辞をしばしば推測できますが、Oracle Developer Studio は、これを明示的に指定することを求めます。明示的に指定すれば、両方を満たすことができます。たとえば、「mov」を「movw」に、「shr」を「shrw」に変更します。
- 「`#`」は、`gcc` アセンブラファイル内ではコメントを導入していますが、Oracle Developer Studio アセンブラファイル内では前処理ディレクティブを導入することが以前から予期されてきました。

詳細については、x86 アセンブリ言語のリファレンスマニュアル (https://docs.oracle.com/cd/E53394_01/html/E54851/index.html)を参照してください

SPARC アセンブラ

正式な SPARC アセンブリ形式は、SPARC アセンブリ言語のリファレンスマニュアル (http://docs.oracle.com/cd/E53394_01/html/E54833/index.html)に定義されています。

ELF セクションに関するアセンブラディレクティブ

.section ディレクティブは SPARC アセンブラでは異なる引数を取ります。属性フラグは、GNU アセンブラ (gas) のような文字列の代わりに、明示的なトークンを使用して定義されます。たとえば、gas の場合、.section ディレクティブは次のように表示されます。

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

Oracle Developer Studio アセンブラまたは Oracle Solaris SPARC アセンブラを使用すると、ディレクティブは次のように表示されます。

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

注記 - SPARC アセンブラは .pushsection ディレクティブと .popsection ディレクティブをサポートしますが、.previous ディレクティブはサポートしません。

擬似オペランドの問題

.symver 擬似オペランドは、GNU 互換性のために SPARC アセンブラでサポートされています。

.uleb128 および .sleb128 の擬似オペランドはサポートされています。

SPARC アセンブラのリソース

詳細は、次のリソースを参照してください。

- 現在の SPARC 命令セットの説明: http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/sparc-processor-2516655.html
- SPARC ABI のドキュメント: http://sparc.org/technical-documents/

ヘッダーファイルの互換性

GCC コンパイラと Oracle Developer Studio コンパイラは異なったシンボルを事前定義します。

gcc が C および C++ のために事前定義したシンボルを確認するには:

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

Oracle Developer Studio C および C++ コンパイラが事前定義したシンボルを確認するには:

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

さまざまなコンパイラオプションを使用すると、すべてのコンパイラについて、事前定義されたマクロとその値に影響を与える可能性があります。-m32|-m64 オプションと言語オプション (std=*v*) は、事前定義されたマクロに特に影響を与えます。

この出力には、主に glibc 内に定義されている、移植性のないさまざまな関数についてのヘッダーファイル宣言を有効にするために使用される _GNU_SOURCE マクロを定義するソースコードも含まれています。詳細な情報は、http://stackoverflow.com/questions/5582211/what-does-define-gnu-sourceimply から入手できます。

プリプロセッサの互換性

Oracle Developer Studio C および C++ コンパイラには、デフォルトで使用される C プリプロセッサの独自のビルトイン実装があります。これらのプリプロセッサには、gcc と互換性のある次の拡張機能があります。

Oracle Developer Studio プリプロセッサ

- #warning
- #include_next (ラッパーヘッダーを実装する)

プリプロセッサの従来モードの動作に依存する一部のコードも、Oracle Developer Studio コンパイラと GCC コンパイラの相違点に遭遇します。gcc の従来モードは、https://gcc.gnu.org/onlinedocs/cpp/Traditional-Mode.html に記載されています。

Oracle Solaris 上で /usr/lib/cpp を使用する従来の動作が必要な場合はその動作を実現できますが、ベストプラクティスは、それらの依存関係をより最新の使用法に置き換えることです。/usr/lib/cpp は、gcc と完全に互換性のあるプリプロセッサとなることを意図したものではありません。

次に示す例は、ソースコードにしばしば見られるものです。

例 1 空のコメントを使用したトークンの結合

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

期待される結果は「FOOBAR」という出力です。これは、従来のモードを有効にする特別な手順を実行しないかぎり、gcc または Oracle Developer Studio のいずれかで動作しません。Oracle Developer Studio C では、これは -Xs オプションを指定することを意味します。gcc では、-traditional-cpp オプションを指定する必要があります。C および C++ で定義されている標準のトークン結合演算子 ## を使用するようにコードを更新することをお勧めします。

例 2 トークン間のスペース

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

ここで期待される結果は「FOO-BAR」で、「FOO - BAR」ではありません。プリプロセッサは「-」記号の前後に空白を追加する場合もあれば、そうしない場合もあります。gcc コンパイラは空白を追加せず、一部のコードは、-E または -P オプションを使用してコンパイラをプリプロセッサとして使用するときに、その動作に依存します。従来の動作を取得するために、Oracle Developer Studio C コンパイラで -Xs を使用するか (Oracle Developer Studio C++ ではサポートされません)、または /usr/lib/cpp を直接使用できます。

コンパイラの互換性

このセクションでは、Oracle Developer Studio と GCC の間の互換性に影響を与えるコンパイラ機能とその他の動作について説明します。

処理系定義の動作

C および C++ の規格の一部は、「処理系定義の動作」になっています。これらの詳細は GCC および Oracle Developer Studio のドキュメントに定義されており、いくつかの点で異なっています。

ビットフィールドと列挙型の両方が、コンパイラの選択によって符号付きまたは符号なしになることがあります。`enum` の場合、この選択は、列挙された値のリストに基づいて異なる可能性があります。Oracle Developer Studio C および C++ の動作は `gcc` の動作と異なります。

符号付きおよび符号なしの `int` ビットフィールド

`int` (`signed int` または `unsigned int` ではない) として宣言されたビットフィールドは、コンパイラによって、符号付きまたは符号なしのいずれかの型を使用して実装される可能性があります。これにより、値を抽出するときに違いが生じ、符号拡張するかどうかの決定にも違いが生じます。

Oracle Developer Studio コンパイラは `int` ビットフィールドに符号なしの型を使用し、`gcc` コンパイラは符号付きの型を使用します。この動作を制御するには、`gcc -funsigned-bitfields` フラグを使用します。

詳細については、`https://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/Non_002dbugs.html` の 6 番目のリスト項目を参照してください。

符号付きまたは符号なしの `enum` 型

C および C++ の `enum` 型は、コンパイラによって、符号付きまたは符号なしのいずれかの型を使用して実装される可能性があります。これは特定の型の列挙値に依存することがあるため、予測がさらに困難です。Oracle Developer Studio コンパイラの動作はこの点で、`gcc` と異なっています。Oracle Developer Studio コンパイラは `enum` に対して符号付きの値を使用し、`gcc` コンパイラは `enum` 型の一部の値に明示的に負の値が代入されていないかぎり、符号なしの型を使用します。

`enum` 型として宣言されているビットフィールド間の実装の差異を理解することは、(今のところは) 読者による練習に頼っています。移植性を高めるようにソースコードを更新することをお勧めします。ビットフィールドについては、コードがどのように記述されているかに応じて、単純に「int」宣言を「signed」または「unsigned」のいずれかに更新します。`enum` 型の場合、型を整数型に変換するときは型をキャストするようにし、そのデフォルトの型に依存しないようにします。

C 言語の拡張

Oracle Developer Studio の C 言語の多くの拡張は Oracle Developer Studio C++ でも実装されます。C 言語の一部の拡張については、『Oracle Developer Studio 12.5: C ユーザーズガイド』の「拡張機能」に記載されています。`gcc` 言語の拡張のリストについては、GCC ドキュメントの C 言語ファミリの拡張についてのセクションを参照してください。

`gcc` テーブルに一覧表示されている項目の一部は、現在は関連する言語規格の標準機能で、表 1「Oracle Developer Studio で実装される GCC の拡張」に C11、C99、C++03、C++11、および C++14 として一覧が記載されています。

新しい言語規格の機能は、その機能が既存のコードと競合するか、厳密な準拠モードを有効にするオプションを使用する場合を除き、古い言語規格によってコンパイルするときに拡張として利用できることがあります。さまざまなモードで使用可能な機能の詳細については、そのコンパイラのドキュメントを参照してください。

次の表に、Oracle Developer Studio で実装されている拡張機能の一覧を示します。

表 1 Oracle Developer Studio で実装される GCC の拡張

GCC の拡張	Oracle Developer Studio の実装ステータス
文の式: 文および宣言を式の内側に入れます。	実装されています。
ローカルラベル: ブロックに対するローカルなラベル。	C でのみ実装されています。
値としてのラベル: ラベルへのポインタの取得と計算された `gotos`。	C でのみ実装されています。
入れ子になった関数: Algol や Pascal のような関数の字句範囲。	実装されていません。
構築呼び出し: 別の関数の呼び出しをディスパッチします。	実装されていません。
typeof: 式のタイプを示しています。	実装されています。Oracle Developer Studio 12.5 C++ の新機能。C++11 はこのための `decltype` を定義します。Sun (`-compat=5`) モードでは、C++ は `typeof` キーワードを省略しますが、`__typeof` および `__typeof__` は引き続きサポートしています。
条件式: 「?:」式の中間のオペランドを省略します。	C でのみ実装されています。
`__int128`: 128 ビット整数—`__int128`。	実装されていません。
Long Long: ダブルワード整数 —`long long int`。	C および C++ で実装されています。
Complex: 複素数を表すデータ型。	型指定のない `_Complex` は `gcc` との互換性のためにデフォルトで `double` に指定されます。C でのみ実装されています。
浮動小数点型: 追加の浮動小数点型。	Oracle Developer Studio C および C++ は 128 ビットの `long double` 型を実装しますが、`__float128` という名前の型を使用しません。
半精度: 半精度浮動小数点型。	実装されていません。
10 進浮動小数点: 10 進浮動小数点型。	実装されていません。
16 進浮動小数点: 16 進浮動小数点定数。	C99.C でのみ実装されています。
固定小数点: 固定小数点型。	実装されていません。
名前付きアドレス空間: 名前付きアドレス空間。	実装されていません。
長さゼロ: 長さゼロの配列 (一般的なゼロ長配列)。	`int foo[0];` C++ で GNU 互換モード、または `-features=zla` で実装されています。
長さゼロ: 長さゼロの配列 (柔軟な配列のメンバー)。	`int foo[]; // 構造体の末尾` C++ で GNU 互換モード、または `-features=zla` で実装されています。
空の構造体: メンバーを含まない構造体。	実装されています。C の場合は `-features=extensions` が必要です。
変数長: 長さが実行時に計算される配列。	C99.C および C++ で実装されています。
可変個引数マクロ: 可変数の引数をとるマクロ	C99.C および C++ で実装されています。Oracle Developer Studio は可変マクロ引数用のユーザー定義名を指定するための `gcc` 拡張を実装します。欠落した可変引数のための `gcc` 拡張は C++ に実装されていません。

GCC の拡張	Oracle Developer Studio の実装ステータス
エスケープされた改行: エスケープされた改行の規則が若干緩やかです。	実装されていません。
添字の指定: すべての配列は `lvalue` でない場合でも添字を指定できます。	C99.標準の C++。C および C++ で実装されています。
ポインタ演算: void ポインタおよび関数ポインタの演算。	C でのみ実装されています。警告が生成されます。
配列へのポインタ: 修飾子を使用した配列へのポインタが予期したとおりに動作します。	実装されていません。
初期化子: 非定数の初期化子。	C99.標準の C++。実装されています。
複合リテラル: 複合リテラルは構造体、共用体、または配列を値として与えます。	C99.C で実装されています。
指示付き初期化子: 初期化子のラベル要素。	C99.C で実装されています。
case の範囲: `` `case 1 ... 9` `` など。	実装されています。
共用体へのキャスト: 共用体のメンバーから `union` 型へのキャスト。	実装されていません。
宣言の混在: 型宣言とコードの混在。	C99.標準の C++。実装されています。
属性の拡張	認識されている属性をテストするために `__has_attribute()` を使用できます。詳細については、10 ページの「属性」を参照してください。
関数プロトタイプ: プロトタイプ宣言と古い形式の定義。	C でのみ実装されています。C++ に関係ありません。
C++ コメント: 認識されません。	C で実装されています。
ドル記号: 識別子でドル記号を使用できます。	実装されています。`-features=iddollar` が必要です。
文字エスケープ: 「¥e」が文字 <ESC> を表します。	実装されていません。
整列: 型または変数の整列について照会します。	C11 では `_Alignof` とスペルします。`__alignof__` は C および C++ でサポートされます。C++ は `alignof` を C++11 モードでサポートします。
インライン化: インライン関数の定義 (マクロと同様の速さ)。	C99 および標準の C++。実装されています。GCC はこの規格を実装する前は、外部の関数本体の代わりに静的な関数本体を作成していました。コードがこれに依存する場合は、C オプション `-features=no%extinl` を使用して、Oracle Developer Studio C コンパイラから類似の動作を取得できます。
Volatile: volatile オブジェクトへのアクセスを構成するもの。	実装されています。GCC と互換性があります。
C とアセンブリ言語の使用: C とアセンブラをインタフェースするための命令と拡張。	実装されています。C および C++ は、制約、`asm()` ラベル、明示的なレジスタ変数などの `gcc` 互換の `asm()` ステートメントを実装します。
別のキーワード: ヘッダーファイルについては `__const__`、`__asm__` など。	実装されています。Oracle Developer Studio 12.5 C++ は `__asm` および `__volatile` キーワード記法を追加しました。
不完全な列挙型: `enum foo;`、後で定義。	C++11。C および C++ で実装されています。C++11 モードである必要があります。
関数名: 現在の関数の名前である出力可能文字列。	Oracle Developer Studio コンパイラは `__func__` および `__PRETTY_FUNCTION__` をサポートします。
戻りアドレス: 関数の戻りアドレスまたはフレームアドレスを取得します。	実装されていません。
ベクトル拡張: 組み込み関数を介してベクトル命令を使用します。	9 ページの「SIMD ベクトルのサポート」を参照してください。
offsetof: `offsetof` を実装するための特殊構文。	実装されていません。
`__sync` 組み込み型: 不可分メモリーアクセス用のレガシー組み込み関数。	実装されています。`-xatomic=studio` を使用します。16 ページの「不可分機能」を参照してください。

GCC の拡張	Oracle Developer Studio の実装ステータス
	(gcc では標準の __atomic 組み込み型が推奨されるため、これらの関数が非推奨とされました。)
__atomic 組み込み型: メモリーモデルを持つ不可分組み込み関数。	実装されています。Oracle Developer Studio 12.5 の新機能。-xatomic=studio を使用します。16 ページの「不可分機能」を参照してください。
整数オーバーフロー組み込み関数: 演算および演算オーバーフロー検査を実行する組み込み関数。	実装されていません。
トランザクションメモリーのための x86 特定メモリーモデル拡張: x86 メモリーモデル。	実装されていません。
オブジェクトサイズ検査: 制限されたバッファーオーバーフロー検査のための組み込み関数。	実装されていません。
ポインタ境界チェッカー組み込み関数: ポインタ境界チェッカー用の組み込み関数。	実装されていません。
Cilk Plus 組み込み関数: Cilk Plus 言語拡張用の組み込み関数。	実装されていません。
その他の組み込み関数: ほかの組み込み関数。	__builtin_constant_p() は、コンパイル時の定数かどうかをテストできます。その他のほとんどの組み込み関数は、Oracle Developer Studio によって実装されていません。
ターゲット組み込み関数: 特定のターゲットに固有の組み込み関数。	実装されていません。
ターゲット形式の検査: 特定のターゲットに固有の形式の検査。	実装されていません。
プラグマ: gcc によって受け入れられるプラグマ。	#pragma once が実装されています。その他は実装されていません。
名前なしフィールド: 構造体/共用体の内部の名前のない構造体/共用体フィールド。	C++11。C および C++ で実装されています。
スレッドローカル: スレッドごとの変数。	C99.C++11。C および C++ で実装されています。
2 進定数:「0b」接頭辞を使用した 2 進定数。	C++ でのみ実装されています。

SIMD ベクトルのサポート

Oracle Developer Studio は、CPU 固有の命令を使用してベクトル演算をエンコードする一部のアーキテクチャー固有の組み込み関数をサポートしています。これらのデータ型は、vector_size 属性を使用するか、または4 ページの「ヘッダーファイルの互換性」に説明されているような適切なヘッダーファイルを含めることによって作成されます。一部の C 演算子はそのような型でサポートされますが、場合によっては CPU 固有の組み込み関数の使用が必要なこともあります。

_m128 などの型とそれらの演算については、『Oracle Developer Studio 12.5: C ユーザーズガイド』の「SIMD 組み込み関数」の SPARC64 X についてのセクションを参照してください。

C++ 固有の機能

Oracle Developer Studio C++ コンパイラは、__cplusplus マクロの非標準の値を、その設定を必要とするソースコードに対して有効にする -features=cplusplus_redef オプションをサポートします。『Oracle Developer Studio 12.5: C++ ユーザーズガイド』の「-xrestrict[=f]」を参照してください。

C++ 用のその他の GNU 拡張は次のとおりです。

- 列挙型の long long 定数
- 関数パラメータとして void の typedef を許可

- `typedef void VOID;`
  `int foo(VOID);`
- `_Bool` - `<stdbool.h>` がインクルードされた場合の Oracle Linux の `bool` と同等です
- `extern` テンプレート
- `long long` ビットフィールド
- `pragma` パックのプッシュ/ポップ

C++ の拡張については、GCC ドキュメントのC++ 言語の拡張に関するセクションを参照してください。Oracle Developer Studio コンパイラ内のこれと同じ機能については、次の表から見つけることができます。

表 2 GNU C++ の拡張

GNU C++ の拡張	Oracle Developer Studio の実装ステータス
C++ Volatile: volatile オブジェクトへのアクセス方法を決定します。	互換性のある実装。
制限付きポインタ: C99 制限付きポインタと参照。	実装されています。
あいまいなリンク: G++ がインラインや仮想テーブルなどを配置する場所。	Oracle Developer Studio はこれらに対し、おそらく若干違った方法で COMDAT を使用します。
C++ インタフェース: 単一の C++ ヘッダーファイルを宣言と定義の両方に使用できます。	実装されていません。`gcc` で非推奨になりました。
テンプレートのインスタンス化: 必要なテンプレートのインスタンス化のコピーがそれぞれ 1 つだけ出力されることを確認するための方法。	Oracle Developer Studio はこれらに対し、おそらく若干違った方法で COMDAT を使用します。
バインドメンバー関数: 「->*」や「.*」式で示すメソッドへの関数ポインタを抽出できます。	実装されていません。
C++ 属性: C++ 専用の変数、関数、および型属性。	10 ページの「属性」を参照してください。
関数の複数バージョン化: 複数の関数バージョンを宣言します。	実装されていません。
名前空間の関連付け: 名前空間の関連付けのためによく使われるディレクティブ。	実装されていません。`gcc` では標準の C++11 機能が推奨されるため、非推奨になりました。
型特性: 型特性のためのコンパイラサポート。	実装されていません。
C++ の概念: 一般的なプログラミングのためのサポートの改善。	実装されていません。
Java 例外: Java を操作するための例外処理の調整。	実装されていません。
非推奨の機能: C++ から除去される予定の機能。	実装されていません。
下位互換性: C++ の以前の定義との互換性。	実装されていません。

属性

Oracle Developer Studio 12.5 コンパイラで実装されている属性の完全なドキュメントは、『Oracle Developer Studio 12.5: C ユーザーズガイド』の「サポートされる属性」および『Oracle Developer Studio 12.5: C++ ユーザーズガイド』の「サポートされる属性」にあります。

`__has_attribute()` 組み込み型は、認識されている属性をテストするために `gcc` および Oracle Developer Studio の両方のコンパイラで使用できます。

Oracle Developer Studio 12.5 C++ の新しい属性は、`aligned`、`deprecated`、`weak(alias)`、`weakref`、`packed`、`tls_model`、`vector_size`、および `visibility` です。

Oracle Developer Studio コンパイラは、属性についてのすべての構文をサポートするわけではありません。

表 3 GCC 属性

GCC 属性カテゴリ	Oracle Developer Studio でサポートされる属性
関数属性: 関数に副次的影響がないこと、または関数が復帰しないことを宣言します。	`alias`、`aligned`、`always_inline`、`const`、`constructor` / `destructor`、`deprecated` (C++14)、`malloc`、`noinline`、`noreturn`、`nothrow` (C++ のみ)、`pure`、`returns_twice`、`visibility`、`weak` (および `alias`)、`weakref` (C++)
変数属性: 変数の属性を指定します。	`aligned`、`deprecated`、`mode` (C++)、`packed` (部分的な実装)、`tls_model`、`vector_size`、`weak`
型属性: 型の属性を指定します。	`aligned`、`deprecated`、`packed`、`visibility`
列挙子属性: 列挙子の属性を指定します。	`deprecated`

コマンド行オプション

Oracle Developer Studio と `gcc` の両方のコンパイラは、`-g`、`-c`、`-o` などの多くの従来のコンパイラオプションを実装します。次の表では、特に `gcc` との互換性を持つように実装されている Oracle Developer Studio コンパイラのオプションについて説明します。

Oracle Developer Studio の次のオプションは、`gcc` と互換性があります。

表 4 互換性のあるオプション

オプション	説明
`-std`	`-std` オプションは、C11 や C++11 などの言語規格を選択します。
`-pedantic`	通常なら受け入れられる標準の技術的な違反に関するエラーまたは警告を発行します。このオプションは Oracle Developer Studio 12.5 C++ コンパイラの新しいオプションです。
`-m32` / `-m64`	32 ビットまたは 64 ビットのバイナリ出力を選択します。
`-shared`	共有ライブラリを生成します。Oracle Solaris Studio 12.4 では、`-shared`オプションは `-G` の別名のように機能していました。Oracle Developer Studio 12.5 では、このオプションは `gcc` の `-shared` オプションのように機能します。表5「違いがあるオプション」を参照してください。

次のオプションは Oracle Developer Studio と GCC の間で動作が異なります。

表 5 違いがあるオプション

Oracle Developer Studio のオプション	GCC のオプション	説明
`-xM`	`-M`	ソースファイルについての make 形式の依存関係を出力します。`gcc` では、`-xM` は別の機能を実行します。Oracle Developer Studio では、`-M` はリンカーのマップファイルを選択します。
`-B`(static| dynamic)	`-Wl,-B`(static| dynamic)	GNU リンカーはこの動作をサポートしますが、`gcc` はサポートしません。オプションを GNU `ld` に渡すには、`-Wl` を使用します。`gcc` では、`-B` は別の機能を実行します。
`-G`	`-shared`	共有ライブラリを生成するときに、`gcc` 内で `-shared` を指定すると、C++ ライブラリの依存関係が追加されます。Oracle Developer Studio コンパイラで `-G` オプションを指定しても、この依存関係は追加されません。

次の gcc 形式のオプションは、Oracle Developer Studio の対応するオプションに自動的に変換されます。このオプションのリストは、このオプション -xhelp=gccflags を C または C++ コンパイラに渡すことで表示されます。

表 6 自動変換されるオプション

Oracle Developer Studio での gcc オプション	Oracle Developer Studio コンパイラの動作
-MM	-xM1 と同じ
-Ofast	-fast と同じ
-Wall	+w2 と同じ (C++ のみ)
-Wall	-v と同じ (C のみ)
-Werror	-errwarn=%all と同じ
-Wpedantic	-pedantic と同じ
-fno-elimininate-unused-debug-types	受け入れて無視
-fopenmp	-xopenmp と同じ
-fPIC	-KPIC と同じ
-fpic	-Kpic と同じ
-fplan9-extensions	受け入れて無視 (C のみ)。
-fplugin-arg-name=*t*	警告と無視 (C のみ)
-fplugin=*t*	警告と無視 (C のみ)
-fsigned-char	-xchar=signed と同じ (C のみ)
-fsyntax-only	-xe と同じ
-funsigned-char	-xchar=unsigned と同じ (C のみ)
-gdwarf-version	-xdebugformat=dwarf と同じ
-gstabs	-xdebugformat=stabs と同じ
-gstabs+	-xdebugformat=stabs の使用が推奨されます。オプションは無視されます
-iplugindir=*t*	警告と無視
-march=*a*	-xtarget=*a* と同じ
-mcpu=*a*	-xtarget=*a* と同じ
-mfmaf	-fma=fused と同じ
-mno-vis	-xvis=no と同じ
-mno-vis2	-xvis=no と同じ
-mno-vis3	-xvis=no と同じ
-mtune=*a*	-xtarget=*a* と同じ
-mvis	-xvis と同じ
-mvis2	-xvis と同じ
-mvis3	-xvis と同じ
-no-canonical-prefixes	受け入れて無視
-no-fma	-fma=none と同じ
-no-fmaf	-fma=none と同じ
-no-trigraphs	-xtrigraphs=no と同じ
-nodefaultlibs	-xnolib と同じ (C++ のみ)
-pass-exit-codes	警告と無視
-pedantic-errors	-pedantic と同じ
-pipe	警告と無視
-save-temps	-keeptmp と同じ

Oracle Developer Studio での gcc オプション	Oracle Developer Studio コンパイラの動作
-shared	-G と同じ
-specs=*t*	警告と無視
-traditional	-Xs と同じ (C のみ)
-trigraphs	-xtrigraphs=yes と同じ

オブジェクトファイルの互換性

このセクションでは、Oracle Developer Studio と GCC の間の互換性に影響を与えるオブジェクトファイルの機能について説明します。

注釈

一部の Oracle Developer Studio ツールは、コンパイラによって出力される生成されたコードに関する追加の構造情報に依存します。この情報は「注釈」と呼ばれ、「.annotations」という名前の ELF セクションに出力されます。コードアナライザの一部の機能は注釈に依存します。このデータの生成は、-xannotate オプションを使用して制御できます。この情報を使用するコマンドは、binopt、code-analyzer、discover、collect、および uncover です。

C++ 符号化名

C++ コンパイラは、型情報が内部にエンコードされる ELF シンボルを生成します。これらは「符号化名」と呼ばれます。符号化名の形式は、実装の詳細であるため、compat=5 (Sun ABI) モードでコンパイルされた Oracle Developer Studio コードは、g++ でコンパイルされたコードと正しく混在できません。gnu モードで Oracle Developer Studio C++ コンパイラを使用すると、互換性のあるオブジェクトファイルになります。詳細については、『Oracle Developer Studio 12.5: C++ ユーザーズガイド』を参照してください。

計測を使用する機能

オブジェクトコードを計測する必要のあるコンパイラ機能は、追加のライブラリまたはオブジェクトファイルをリンク時に含める必要があります。これらの機能を使用する場合、Oracle Developer Studio コンパイラを使用して、実行可能ファイルまたはライブラリをリンクする必要があります。gcc でビルドされたコードをプログラムまたはライブラリに混在させた場合は計測されないため、結果が不完全になります。

このカテゴリの機能は次のとおりです。

- プロファイルフィードバック – オプション -xprofile および -xlinkopt を含みます。
- 従来のプロファイリング – コンパイルおよびリンク時にオプション -xpg を含みます。
- 従来のカバレージ – オプション -xprofile=tcov および tcov ユーティリティを含みます。

デバッグ情報

dbx およびその他の Oracle Developer Studio ツールは、コンパイラによって生成されるデバッグ情報を使用します。情報の大部分が DWARF 形式で記録されますが、いくつかの追加のインデックス情報は

STABS と呼ばれる古い形式で記録されます。dwarf の ELF セクションは「.debug」で始まり、stabs の ELF セクションは「.stab」で始まります。

実行時サポートの互換性

このセクションでは、実行時サポートに関連する問題について説明します。

libgcc のサポート

コンパイラは必要に応じて libgcc を自動的にリンクしますが、何か不具合があった場合に、ライブラリが何をするためのものかを知っておくと便利なことがあります。gcc サポートライブラリ (libgcc) には、libgcc.a と呼ばれるアーカイブライブラリと libgcc_s.so と呼ばれる共有ライブラリの形式の、さまざまなヘルパールーチンがあります。libgcc には、例外をサポートするルーチンと、現在のアーキテクチャーでは直接的な命令がない場合がある整数と浮動小数点演算をサポートするルーチンが含まれています。ライブラリの機能については、gcc ドキュメントのGCC 低レベル実行時ライブラリに関するセクションに記載されています。

gcc コンパイラを使用して作成される実行可能ファイルは、libgcc.a に対してリンクされ、libgcc_s.so に対する動的依存関係はありません。g++ コンパイラを使用して作成される実行可能ファイルは代わりに libgcc_s.so を使用し、これに対する動的依存関係があります。

Oracle Developer Studio の C コンパイラによって生成されたコードは libgcc に対する依存関係がありません。

Oracle Developer Studio C++ コンパイラによって生成されたコードは Sun モード (-compat=5) でビルドされる場合は libgcc に依存しませんが、GNU 互換モード (-compat=g-std=c++03、-std=c++11、または -std=c++14) でビルドされる場合はそのような依存関係を持ちます。

Oracle Linux の場合、libc_supp.a という名前の類似のライブラリが Oracle Developer Studio に含まれています。

C++ 実行時のサポート

Oracle Developer Studio C++ コンパイラによってコンパイルされたソースファイルは、正しい実行時ライブラリ、CRT ファイル、およびリンカーオプションが使用されるようにするために、Oracle Developer Studio C++ コンパイラを使用してリンクされる必要があります。g++ オブジェクトファイルは g++ でリンクする必要があります。互換性のない多くの実装の詳細がオブジェクトファイル内で使用されているため、Oracle Developer Studio C++ コンパイラおよび g++ コンパイラからのオブジェクトファイルを同じ実行可能ファイルまたは共有ライブラリに混在させることはサポートされていません。

GNU ABI の互換性

Oracle Developer Studio C++ によって GNU 互換モード (オプション -compat=g、-std=c++03、-std=c++11、または -std=c++14 を使用して) で作成された共有ライブラリは、g++ コンパイラによって作成された共有ライブラリと混在させて、いずれかのコンパイラによって作成された主プログラムにリンクさせることができますが、これらは同じバージョンの g++ ABI を使用する必要があります。

GNU 互換モードで動作するとき、Oracle Developer Studio C++ コンパイラは、ライブラリインタフェース、名前符号化、および標準ライブラリオブジェクトのバイナリレイアウトの領域で、g++ コードとバイナリ

レベルで互換性があります。ただし、実装のほかの多くの基本的な側面では、別のメカニズムを使用します。外部インライン関数、テンプレートインスタンス、RTTI レコード、および例外範囲情報は、COMDAT を使用して異なる方法で実装されます。例外範囲情報は Studio 固有の方法でフォーマットされます。このため、Oracle Developer Studio C++ でコンパイルされたオブジェクトファイルやアーカイブライブラリと、`g++` でコンパイルされたオブジェクトファイルやアーカイブライブラリの間で実行できる混在の量が制限されます。

GNU C++ ABI 5.*x* の変更内容

GNU C++ ABI のバージョン 4.*x* と 5.*x* の間の互換性のない変更により、`g++` でコンパイルされた実行可能ファイルとライブラリが正しく動作するには、これらが 1 つの ABI を一貫して使用する必要があります。5.*x* GNU C++ ライブラリは両方の ABI をサポートしますが、コンパイラ (GCC または Oracle Developer Studio) は一度に 1 つの ABI のみを生成できます。`g++` コンパイラは、デフォルトで 5.*x* ABI を指定します。4.*x* ABI を選択するには、`-D_GLIBCXX_USE_CXX11_ABI=0` を指定してコンパイルします。

Oracle Developer Studio 12.5 C++ コンパイラは 4.*x* ABI のみを実装します。共有ライブラリと実行可能ファイルを混在させるには、上記のオプションを使用して `g++` 部分をビルドします。この問題についての `g++` のドキュメントは、https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html にあります。

技術的に、4.*x* ライブラリまたは実行可能ファイルが 5.*x* ライブラリまたは実行可能ファイルと一緒に使用されるのを止める方法はありませんが、これらは 2 つの ABI の異なるライブラリクラスを使用して相互に呼び出すことはできません。これらのクラスの一部は (文字列クラスのように) 一般的によく使用されるため、2 つの ABI を混在させてもたいした効果は得られません。

`main` 関数

C++ の `main` 関数にはいくつかの追加メカニズムがあります。いずれのコンパイラを使用してオブジェクトファイルをコンパイルする場合であっても、実行可能ファイルをリンクするために `main` も使用するべきです。コンパイラドライバによって使用される crt オブジェクトファイルは、オブジェクトファイル `main` に組み込まれたメカニズムと一致します。

libCrunG3.so ライブラリ

GNU モードでコードをコンパイルするとき、Oracle Developer Studio C++ コンパイラは GNU C++ ライブラリを使用しますが、さらに RTTI のサポート機能、例外、およびその他の言語機能を提供するために、`libCrunG3.so` と呼ばれる追加ライブラリをリンクする必要があります。このライブラリは、Sun モード (`-compat=5`) で使用される `libCrun.so` ライブラリに対応します。

異なるバージョンの GNU C++ ライブラリ

Oracle Developer Studio 12.5 C++ コンパイラ (GNU 互換モード) は 5.1.*x* バージョンの GNU C++ ライブラリを使用します。異なるバージョンの GNU C++ ライブラリを使用してビルドされた共有ライブラリおよびアプリケーションを混在させることができますが、メインアプリケーションをビルドおよびリンクするために使用されるコンパイラは、いずれかの共有ライブラリを作成するために使用されるコンパイラと同じかそれより新しいバージョンである必要があります。いずれかの共有ライブラリが Oracle Developer Studio C++ でビルドされた場合、メインアプリケーションは、Oracle Developer Studio C++ コンパイラまたは 5.1.*x* GNU C++ ライブラリをサポートする `g++` コンパイラのいずれかを使用してビルドされる必要があります。

実行時ライブラリの場所

C++ 実行時ライブラリはさまざまな場所にあります。

次の Sun モード (`-compat=5`) C++ ライブラリは、Oracle Solaris では `/usr/lib` にありますが、Oracle Linux にはありません。

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

次の GNU モード C++ ライブラリは、Oracle Solaris では `/usr/lib` にありますが、Oracle Linux にはありません。

- `libCrunG3.so` (ただし、Oracle Solaris パッケージ `SUNWlibC` のパッチが必要になることがあります)
- `libdemangle.so`

ほかのすべての C++ 実行時ライブラリはコンパイラのインストールディレクトリにあります。

Oracle Developer Studio C++ コンパイラは、Oracle Developer Studio にバンドルされているすべての共有ライブラリに対してリンクするとき、通常は、実行時にライブラリを見つけるために必要な `RPATH` 設定を実行可能ファイルに追加します。この動作を防ぐことは可能で、`-xnorunpath` オプションを追加して適切な `-R` オプションを追加することで、`RPATH` 設定を自分で構成できます。

OpenMP

ソースコードを `-xopenmp` でコンパイルすると、システムライブラリへの実行時依存関係が生成され、これは Oracle Developer Studio コンパイラの場合は `libmtsk` ライブラリです。Oracle Solaris では、このライブラリは `/usr/lib` にあります。Oracle Linux では、このライブラリはコンパイラのインストールディレクトリにあります。スレッド管理はプロセス全体の動作であるため、1 つのプログラム内で使用する OpenMP 実装は 1 つのみとしてください。共有ライブラリが OpenMP を使用する場合、メインアプリケーションは異なる OpenMP 実装を使用できません。OpenMP の Oracle Developer Studio 実装は、OpenMP の GCC 実装との互換性がありません。

不可分機能

不可分機能は、オペレーティングシステムからの実行時サポートを必要とする C 2011 および C++ 2011 規格の新しい言語機能です。Oracle Developer Studio コンパイラは、一部のバージョンの GCC 実行時ライブラリと互換性のある事前バンドル済み実行時ライブラリによってこの機能をサポートするため (『Oracle Developer Studio 12.5: C++ ユーザーズガイド』の「バンドルされている不可分ライブラリ」を参照)、`-xatomic` オプションを使用して、プログラムとライブラリをリンクするときに使用する実行時ライブラリを選択できます。

スレッドローカル

GCC コンパイラおよび Oracle Developer Studio コンパイラは、OS ABI (Oracle Solaris または Oracle Linux) に準拠する方法でスレッドローカルデータ (`__thread declarations`) を実装するため、Oracle Developer Studio コンパイラおよび GCC コンパイラによってコンパイルされたコードの

機能には互換性があります。スレッドローカルデータについての Oracle Solaris ABI の詳細については、『Oracle Solaris 11.3 リンカーとライブラリガイド』を参照してください。

共有ライブラリの互換性

Oracle Developer Studio C++ コンパイラによってコンパイルされたソースファイルは、正しい実行時ライブラリおよびリンカーオプションが使用されるようにするために、C++ コンパイラを使用してリンクされる必要があります。G++ オブジェクトファイルは `g++` コンパイラでリンクされるべきです。

Oracle Developer Studio CC によってオプション `-compat=g`、`-std=c++03`、`-std=c++11`、または `-std=c++14` を使用して作成された共有ライブラリは、`g++` コンパイラによって作成された共有ライブラリと自由に混在させて、いずれかのコンパイラによって作成された主プログラムとリンクさせることができます。ほとんどの場合、`gcc` ライブラリの新しいバージョンに対してコンパイルされたバイナリは、古いバージョンの `gcc` ライブラリにリンクできません。

`gcc` 互換モードの Oracle Developer Studio 12.5 C++ コンパイラは、5.1.*x* バージョンの `g++` 実行時ライブラリを使用します。

異なるバージョンの `g++` ライブラリを使用してビルドされたライブラリおよびアプリケーションを混在させることができますが、メインアプリケーションをビルドおよびリンクするために使用されるコンパイラは、いずれかの共有ライブラリを作成するために使用されるコンパイラと同じかそれより新しいバージョンである必要があります。いずれかの共有ライブラリが Oracle Developer Studio C++ でビルドされた場合、メインアプリケーションは、Oracle Developer Studio C++ コンパイラか、または 5.1.*x* 実行時ライブラリをサポートする `g++` コンパイラのいずれかを使用してビルドされるべきです。

ツールの互換性

このセクションでは、Oracle Developer Studio のツールと GCC の互換性について説明します。

dbx デバッガ

`dbx` は、Oracle Developer Studio コンパイラおよび GCC コンパイラによって生成される C および C++ コードの両方をデバッグできます。これは、18 ページの「GCC バージョンの互換性」で一覧表示されている GCC のバージョンと互換性があります。

パフォーマンスアナライザ

パフォーマンスアナライザは、Oracle Developer Studio バイナリと GCC バイナリが混在した状態をサポートします。これは、Oracle Developer Studio コンパイラおよび GCC コンパイラによってコンパイルされたコードを分析でき、18 ページの「GCC バージョンの互換性」に一覧表示されているバージョンの GCC をサポートしています。

コードアナライザ

Discover ADI (`-i`*adi*) および Discover Lite (`-l`) は、Oracle Developer Studio バイナリおよび `gcc` バイナリをサポートします。`discover(1)` ユーティリティーは、Oracle Developer Studio でコンパイルさ

れたコードのエラーを検出しますが、gcc でコンパイルされたコードを検査しません。uncover(1) ユーティリティーは、Oracle Developer Studio でコンパイルされたコードのカバレージ情報を収集します。

IDE

IDE プロジェクトは、GCC または Oracle Developer Studio のどちらか 1 つのコンパイラで記述またはビルドされたコードのみをサポートします。この制限の回避方法は、GCC で記述するコードについて別々のプロジェクトを作成することです。

GCC バージョンの互換性

GCC バージョン 4.8.2、4.9.2、および 5.1.0 は、次のオペレーティングシステムでサポートされています。

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

GCC バージョン 4.4 は Oracle Linux 6.6 でサポートされています。

詳細情報の参照先

- C 言語ファミリの拡張 (https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html)
- C++ 言語の拡張 (https://gcc.gnu.org/onlinedocs/gcc/C_002b_002b-Extensions.html#C_002b_002b-Extensions)
- GCC バイナリの互換性 (https://gcc.gnu.org/onlinedocs/gcc/Compatibility.html)
- GNU C++ ABI ポリシーおよびガイドライン (https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html)
- Intel® Compilers for Linux: GNU コンパイラとの互換性 (https://software.intel.com/en-us/articles/intel-compilers-for-linux-compatibility-with-gnu-compilers)

Oracle Developer Studio 12.5: GCC 互換性ガイド

Part No: E71994

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

Part No: E71994

ORACLE