Oracle® Solaris Studio 12.4: リリースノート

Part No: E57202
2015 年 5 月

ORACLE®

Part No: E57202

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。

目次

このドキュメントの使用方法

- **概要** – この Oracle Solaris Studio 12.4 リリースでの、システム要件および既知の問題やその他の問題に関する情報を提供します。
- **対象読者** - アプリケーション開発者、システム開発者、アーキテクト、サポートエンジニア。
- **必要な知識** - プログラミング経験、ソフトウェア開発テスト、ソフトウェア製品を構築およびコンパイルできる能力。

製品ドキュメントライブラリ

コンパイラおよびツールの詳細については、関連するマニュアルページまたはヘルプ、および http://docs.oracle.com/cd/E37069_01 にあるドキュメントライブラリを参照してください。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

Oracle Solaris Studio 12.4 リリースノート

このドキュメントでは、この製品をインストールおよび使用する前に知っておく必要がある情報を提供します。インストール手順については、『Oracle Solaris Studio: 12.4 インストールガイド』を参照してください。

システム要件

Oracle Solaris Studio 12.4 ソフトウェアは、SPARC ベースまたは x86 ベースのプラットフォームの Solaris 10 または Oracle Solaris 11.2 オペレーティングシステム、または Oracle Linux オペレーティングシステムにインストールできます。

表 1 システム要件

システムリソース	要件
ハードウェア	64 ビット SPARC および x86 プラットフォーム
オペレーティングシステム	Oracle Solaris 10 8/11 または 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 および 3 必要なパッケージがシステムにあることを確認するには、10 ページの「必要なシステムソフトウェアパッケージ」を参照してください。
配布	Oracle Solaris 10 の場合、SVR4 パッケージのインストール Oracle Solaris 11.2 の場合、IPS パッケージのインストール Oracle Linux の場合、RPM パッケージのインストール Tar ファイル (すべてのプラットフォーム) すべての配布のインストール手順については、『Oracle Solaris Studio: 12.4 インストールガイド』を参照してください。
VM サポート	Solaris 10 または Solaris 11 で、非大域ゾーン、LDOM、またはカーネルゾーンとして Oracle Solaris Studio を実行している Oracle Solaris 11.2 大域ゾーン

システムリソース	要件
サポートされるクライアントシステム リモートで使用するために、IDE またはパフォーマンスアナライザの配布をインストールする場合	IDE、パフォーマンスアナライザ、コードアナライザ、および `dbxtool` を実行するクライアントシステムには、最低限の Java バージョンがインストールされている必要があります
Java バージョン	Java 7、最低限のバージョン 1.7.0_25 Java 8

必要なシステムソフトウェアパッケージ

Oracle Solaris Studio 12.4 リリースをインストールする前に、ソフトウェアをインストールするシステムに、Oracle Solaris Studio ソフトウェアを正常に実行するために必要なシステムパッケージがあることを確認します。次の表は、サポートされている各オペレーティングシステムに必要なソフトウェアパッケージを一覧表示しています。

表 2 Oracle Solaris Studio で必要なシステムソフトウェア

オペレーティングシステムのバージョン	システムソフトウェアパッケージ
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	pkg://solaris/consolidation/sunpro/sunpro-incorporation の更新に関する重要な情報については、『Oracle Solaris Studio: 12.4 インストールガイド』の「Oracle Solaris Studio 12.4 が必要とする Oracle Solaris 11 システムライブラリの更新」を参照してください。 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

オペレーティングシステムのバージョン	システムソフトウェアパッケージ
	pkg:/system/library/math pkg:/system/library/openmp pkg:/system/linker pkg:/system/resource-mgmt/resource-pools
Oracle Linux 5.8 - 5.10 Oracle Linux 6.0 - 6.5	開発/ライブラリパッケージグループおよび次のライブラリを含む 32 ビットルーチンサポートライブラリ: glibc glibc.i686 glibc-devel glibc-devel.i686 elfutils-libelf-devel elfutils-libelf-devel.i686 zlib zlib.i686 libstdc++ libstdc++.i686 libgcc libgcc.i686

インストール手順

すべての配布のインストール手順については、『Oracle Solaris Studio: 12.4 インストールガイド 』を参照してください。

今回のリリースで削除された機能

次の機能は、Oracle Solaris Studio 12.4 ソフトウェアから削除されました。

DLight プロファイリングツール

DLight は、Oracle Solaris Studio 12.3 および 12.2 で提供されていたグラフィカルプロファイリングツールです。Oracle Solaris Studio IDE を使用すると、DLight の一部であったツールと同じいくつかのツールを使用してプロジェクトをプロファイルできます。

使いやすくなったパフォーマンスアナライザを使用して、アプリケーションをプロファイルすることもできます。

C++ Rogue Wave `tools.h++` ライブラリ

`tools.h++` は、Oracle Solaris Studio および Sun Studio ソフトウェアの以前のリリースで提供されていた C++ ファウンデーションクラスライブラリです。このライブラリは 1996 年にリリースされてから、大幅な更新は行われていません。時間や日付のクラスには、修正不可能な深刻なバグが含まれています。

tools.h++ の機能は、Oracle Solaris Studio で提供されている C++ 標準ライブラリ、およびオープンソースの BOOST ライブラリの別のプログラミングインタフェース (API) を使用して利用できます。BOOST ライブラリについては、http://www.boost.org を参照してください。

systent.h ヘッダーファイル

この C++ ヘッダーファイルは Cfront の名残であり、unistd.h よりも古いファイルです。代わりに unistd.h を使用してください。

Fortran の古いモジュール形式

Forte Developer 7 Fortran 95 7.0 以前のリリースで生成されたモジュール形式は、サポートされなくなりました。このリリースの Fortran コンパイラでは、これらの古いリリースで作成された .mod ファイルを読み取ることができません。

コンパイラオプション -xbinopt=prepare

-xbinopt=prepare オプションは廃止されました。代わりに、-xannotate=yes を使用してください。

コンパイラオプション -xcrossfile

-xcrossfile オプションは廃止されました。代わりに、-xipo オプションを使用してください。

アセンブラの -xarch オプション

Oracle Solaris x86 アセンブラの fbe コマンドのオプション -xarch=amd64 および -xarch=generic64 はサポートされなくなりました。

代わりに、-m64 オプションを使用してください。

登録機能

Sun Studio ユーザーが Sun Microsystems に登録することによって最新の製品情報を受け取ることができる登録機能は、2011 年に廃止されました。この機能は Oracle Solaris Studio 製品から削除されました。

今後のリリースで削除される可能性のある機能

次の機能は、今後のリリースで削除される可能性があります。

コンパイラの -xarch オプション

すべての Oracle Solaris Studio コンパイラで、コマンドオプション `-xarch=amd64` および `-xarch=generic64` は廃止され、今後のリリースで削除される可能性があります。

代わりに、`-m64` オプションを使用してください。

C コンパイラオプション -Xs

`-Xs` オプションは、今後のリリースで削除される可能性があります。正しく構築してコンパイルするために C コードで `-Xs` が必要な場合は、少なくとも ISO C 標準の C99 の文法に準拠するようにコードを移行してください。つまり、`-std=c99` オプションでコンパイルできるようにします。

lint ユーティリティーの -Nlevel オプション

拡張された解析のレベルを指定するための `lint` ユーティリティーの `-Nlevel` オプションは、今後のリリースで削除される可能性があります。

lock_lint ユーティリティー

マルチスレッドプログラムのロックを解析する `lock_lint` ユーティリティーは廃止され、今後のリリースで削除される可能性があります。

libgc ライブラリ

Studio の `libgc` ライブラリは廃止されたガベージコレクションライブラリであり、今後のリリースで削除される可能性があります。Oracle Solaris `libgc` ライブラリを使用してください。

パフォーマンスライブラリ内のオプションの引数を含む F95 インタフェース

libsunperf F95 インタフェースのオプションの引数機能は、今後のリリースで削除される可能性があります。オプションの引数のない F95 インタフェースは引き続きサポートされます。

VMS 互換ライブラリ libV77

libV77 ライブラリには Fortran 90 組み込みルーチンと競合し、4 桁の年の Y2K 要件に準拠していない 2 つのルーチンが含まれています。それらは VMS 互換性のためにのみ、オプションのルーチンとして提供されており、今後のリリースで削除される可能性があります。

コンパイラオプション -xanalyze=code

-xanalyze=code オプションは非推奨であり、今後のリリースで削除される可能性があります。代わりに -xprevise オプションでコンパイルし、コードアナライザを使用して表示できるソースコードの静的解析を生成するようにしてください。

libsunmath の静的バージョン

Studio 数学ライブラリの静的バージョン libsunmath.a は、今後のリリースで削除される可能性があります。動的バージョンの libsunmath.so は、引き続き使用できます。オプション -Bstatic -lsunmath を -Bdynamic -lsunmath で置き換えることによって、静的ライブラリの使用を動的ライブラリで置き換えます。

従来の C++ iostream

従来型の iostream (libiostream) は iostream の 1986 オリジナルバージョンであり、1998 C++ 標準で置き換えられました。このライブラリは -library=iostream オプションを使用して指定します。従来型の iostreams は標準ではなく、このライブラリの 2 つの実装は同じではないため、それを使用するコードは移植不可能で、C++ 標準ライブラリと互換性がありません。C++ 標準ライブラリで提供されている iostream 機能を使用してください。

レガシー SPARC ベースシステムのコンパイラオプション

レガシーシステムのサポートは、今後のリリースで削除される可能性があります。これらのシステムには、UltraSPARC I、II、IIe、III、IIIi、III+、IV、および IV+ プロセッサを基にしたシステムが含まれます。

このため、cc、CC、および f95 コンパイラコマンドの次のオプションが削除される可能性があります。

```
-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}
```

コンパイラオプション -xdebugformat=stabs

すべてのコンパイラの -xdebugformat=stabs は、今後のリリースで削除される可能性があります。唯一のデバッガ形式オプションは、現在デフォルトである -xdebugformat=dwarf となります。

dbx での実行時チェック

dbx の実行時検査 (RTC) 機能は削除される可能性があります。実行時にメモリーチェックを行う場合は、discover ツールを使用できます。

レガシー Fortran F77 オブジェクトファイル

レガシー F77 コンパイラおよび -lf77compat リンクオプションを使用して作成されたオブジェクトファイルのサポートは打ち切られる可能性があります。

Fortran のレガシー配列組み込み実行時ライブラリ

SPARC プラットフォームのライブラリ
libfmaxlai、libfmaxvai、libfminlai、libfminvai、libfprodai、および libfsumai は、2005

年に Sun Studio 10 がリリースされてから Studio Fortran コンパイラによって使用されていません。

これらのライブラリは今後のリリースで削除される可能性があります。その時点で、Sun Studio 10 リリースより前の Studio コンパイラによって生成されたオブジェクトファイルおよび実行可能ファイルは使用できなくなり、新しい Studio コンパイラで再コンパイルする必要があります。これらのライブラリが必要な古いオブジェクトファイルおよび実行可能ファイルがあり、再コンパイルできない場合は、古いコンパイラのインストール環境を維持するか、必要な特定のライブラリを古いコンパイラのインストール環境から新しいコンパイラのインストール環境にコピーしてください。

`collect -R` オプション

`collect -R` オプションは、以前はパフォーマンスアナライザツールの新機能の README ファイルを表示するために使用されていました。この情報は、『Oracle Solaris Studio 12.4 リリースの新機能 』の第 3 章「パフォーマンス解析ツール」で公開されています。

サードパーティーソフトウェアの情報

Oracle Solaris Studio 12.4 ソフトウェアには、『Oracle Solaris Studio 12.4: Third Party Notices and Licenses 』によって管理されるサードパーティーのテクノロジが含まれています。

Usage Data to Oracle

使用状況をオラクルに送付する機能は、Oracle Solaris Studio コンポーネントの使用状況に関する情報を、Oracle Corporation に定期的に送信します。この情報は、Oracle Solaris Studio ソフトウェアの今後のリリースを改善する目的で、Oracle Corporation によって使用されます。この情報は匿名であり、この情報を個人や組織に関連付けることは一切できません。

ただし、Usage Data to Oracle を無効にする場合は、`SUNW_NO_UPDATE_NOTIFY` 環境変数をゼロ (`0`) 以外の任意の値に設定してください。

このリリースのドキュメント

この Oracle Solaris Studio 12.4 リリースでは、次のドキュメントを利用できます。

- 『Oracle Solaris Studio: 12.4 インストールガイド 』。このドキュメントには、tar ファイル、グラフィカルパッケージインストーラ、およびコマンド行パッケージインストーラからインストールするための情報があります。
- 『Oracle Solaris Studio 12.4 リリースの新機能 』。このドキュメントは、Oracle Solaris Studio 12.4 リリースの新機能について説明しています。
- 『Oracle Solaris Studio 12.4: 概要 』。このドキュメントでは、コンパイラやツールを使用する方法、およびそれらを連係させて使用する方法について一般的な説明をしています。
- オンラインヘルプ。IDE の「ヘルプ」メニュー経由で使用可能なオンラインヘルプは、IDE のすべてのコンポーネントの使用方法に関するタスク指向の情報を提供します。パフォーマンスアナライザ、スレッドアナライザ、コードアナライザ、および `dbxtool` のオンラインヘルプは、各ツールの「ヘルプ」メニュー経由で使用できます。
- マニュアルページ。ユーザーコマンド、コンパイラに付属しているライブラリ、およびその他のタイプのコマンドを説明するオンラインのリファレンスマニュアルページです。コマンドの構文、使用法、関連コマンドなど、参考情報が含まれています。

  これらのドキュメントには、インストールされた Oracle Solaris Studio 12.4 ソフトウェアの `man` コマンドを使用してアクセスできます。
- Oracle Solaris Studio 12.4 のマニュアルやチュートリアル。これらのドキュメントは、ドキュメントライブラリ (`http://docs.oracle.com/cd/E37069_01/`) から HTML 形式または PDF 形式でアクセスできます

このリリースでの既知の問題、制限事項、および回避策

このセクションでは、この Oracle Solaris Studio 12.4 リリースの時点でのいくつかの既知の問題、およびこれらの問題を回避する方法について説明します。

コンパイラの問題

ここでは、このリリースでのコンパイラに関する既知の問題および回避策について説明します。

コンパイラに共通する問題

このセクションでは、`cc`、`CC`、および `f95` コンパイラに当てはまる既知の問題について説明します。

予約済みプロセッサを使用した、OpenMP プロセッサのバインドに関する問題

Linux の OpenMP プロセッサのバインドでは、`taskset` コマンドを使用したプロセッサ予約が考慮されません。

C++ コンパイラの問題

このセクションでは、このリリースでの C++ コンパイラに関する既知の問題および回避策について説明します。

C++11 標準ヘッダー

次のヘッダーは、同時実行機能用であり、Oracle Solaris Studio 12.4 リリースではサポートされません。

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

BOOST ライブラリのコンパイルに関する問題

すべての BOOST ライブラリが正常にコンパイルされるとは限りません。コンパイルの結果は、使用している BOOST のバージョン、およびライブラリの使用方法によって異なります。BOOST ライブラリは Oracle Solaris Studio コンパイラのテストに含まれており、すべての BOOST をコンパイルできるように対応が進められています。作業の進行を妨げる問題が発生した場合はそれを報告してください。

Oracle Solaris での Apache 標準ライブラリの問題

Oracle Solaris 10 8/11 以降、および Oracle Solaris 11、11.1、11.2 の初期リリースにインストールされている Apache `stdcxx` ライブラリは、新しいコンパイラのデフォルトである `-template=no%extdef` を指定すると正しく動作しません。このライブラリを使用する `CC` コマンド行にオプション `-template=extdef` を追加する必要がある可能性があります。このライブラリの問題の修正は、今後の Oracle Solaris のアップデートで利用可能になります。詳細は、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`) を参照してください。

Oracle Solaris 10 8/11 にインストールされた Apache `stdcxx` ライブラリでは、ヘッダー `stdcxx4/loc/_moneypunct.h` で構文エラーが発生します。このエラーは、以前のコンパイラでは見られなかったものですが、Oracle Solaris Studio 12.4 C++ コンパイラでは検出されます。このエラー検出を無効にする方法はありません。

注記 - この構文エラーの問題は、Oracle Solaris 10 1/13 および Oracle Solaris 11.1 で修正されました。

あいまいさ: コンストラクタ呼び出しまたは関数へのポインタ

C++ では、あるときは宣言と解釈されたり、またあるときは式と解釈される可能性がある文があります。C++ のあいまい排除規則では、ある文を宣言文とみなすことができる場合は、その文は宣言文とすることになっています。

Oracle Solaris Studio 12.2 以前のコンパイラでは、次のような事例を誤って解釈していました。

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

このプログラマはおそらく、最後の行で S 型の一時的な値で初期化される変数 v を定義するつもりでした。従来のバージョンのコンパイラは、この文をそのように解釈していました。

しかし、宣言コンテキスト内のコンストラクト、"S()" は、"S 型の値を戻すパラメータのない関数" を意味する抽象宣言子 (識別子のない抽象宣言子) とみなすこともできます。この事例では、関数ポインタ、"S(*)()" に自動的に変換されています。この文はまた、戻り値が T 型で、パラメータが関数ポインタ型の関数 v の宣言としても有効です。

現在ではコンパイラが正しい解釈をするようになったので、このプログラマが意図したようにならない可能性があります。

あいまいにならないようにコードを修正するには、次の 2 通りの方法があります。

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

1 行目の 1 対の余分な括弧は、v1 の構文が関数宣言としては不正であるので、"S 型の一時的な値で初期化される T 型のオブジェクト" という意味にしか解釈できません。

同様に、コンストラクト "S(*)()" は値とは考えられないので、関数宣言の意味にしか解釈できません。

1 行目は次のように記述することもできます。

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

意味は完全に明確になりますが、この初期設定の形式では、通常はそうでもないとはいえ、一時的な値として非常に大きな値が生成されることがあります。

次のようにコーディングするのはお勧めできません。その理由は、意味が不明確で、コンパイラが異なると結果が異なる可能性があるからです。

```
T v( S() ); // 推奨しない
```

リンク時の名前符号化の問題

次の状態では、リンクの問題が発生することがあります。これは、デフォルトの -compat=5 モードを使用した場合にのみ当てはまります。

- const パラメータ付きで宣言されている関数が、別の場所で const パラメータなしで宣言されている。

  例:

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

  これらの宣言は等価ですが、コンパイラは異なる符号化名を付けます。この問題を回避するには、値のパラメータを const として宣言しないでください。たとえば、関数定義の本体などのあらゆる場所で void foo1(int); を使用します。
- 関数に同じ複合型のパラメータが 2 つあり、一方のパラメータだけ typedef で宣言されている。

  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*);
  ```

  すべての foo2 宣言は等価で、これらは同じものを符号化する必要があります。しかし、コンパイラは一部に異なった符号化を行なっています。この問題を回避するには、一貫して typedef を使用します。

  typedef を一貫して使用できない場合は、回避策として、関数を定義しているファイルに weak シンボルを使用し、宣言とその定義を等価にします。例:

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

  ターゲットアーキテクチャーによって異なる符号化名があります。たとえば、size_t は SPARC V9 アーキテクチャー (-m64) では unsigned long ですが、それ以外のアーキテクチャーでは unsigned int です。これは、2 つの異なったバージョンの符号化名がそれぞれ 1 つのモデルに存在するケースです。このような場合は、2 つのプラグマを用意し、適切な #if 指令で制御する必要があります。

  -compat=g または -std=c++11 を指定してコンパイルした場合、このセクションの内容は当てはまりません。

大域的ではない名前空間のオブジェクトをテンプレートから参照できない

プログラムでテンプレートと静的オブジェクトを使用していると、-instances=extern を指定してコンパイルした場合に未定義シンボルのリンク時エラーが発生します。これは、デフォルト設定の -instances=global では問題になりません。コンパイラは、大域的でない名前空間スコープのオブジェクトに対するテンプレートからの参照をサポートしません。次の例を考えてみましょう。

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

この例では、テンプレートクラスのメンバーは静的な名前空間スコープ変数を参照します。名前空間スコープはファイルスコープを含むことに注意してください。コンパイラは、静的な名前空間スコープ変数を参照するテンプレートクラスのメンバーをサポートしません。複数のコンパイル単位からテンプレートがインスタンス化されると、各インスタンスは異なる k を参照します。つまり、C++ 単一定義規則違反が発生し、コードは定義されていない動作を起こします。

k の使用方法やその効果に応じて、次の代替案が考えられます。2 番目のオプションは、クラスのメンバーの関数テンプレートにのみ使用できます。

1. 変数に外部リンケージを持たせる

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

   すべてのインスタンスは、k の同じコピーを使用します。
2. 変数をクラスの静的メンバーにする

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

   静的なクラスメンバーは外部リンケージを持ちます。C<T>::foo のインスタンスが使用する k はそれぞれ異なります。C<T>::k のインスタンスは、ほかの関数で共有することができます。通常はこのオプションが使用されます。

名前空間内の #pragma align と符号化名

名前空間内で #pragma align を使用する場合は、符号化名を使用する必要があります。たとえば、次のコードでは、#pragma align 文は何の働きもしません。この問題を解決するには、#pragma align 文の a、b、および c を符号化された名前に変更します。

```
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 コンパイラの問題

このリリースの cc コンパイラでは、次の問題に注意する必要があります。

-xustr=ascii_utf16_ushort オプションは -std=c11 と互換性がない

-std=c11 (コンパイラのデフォルト) が有効である場合、フラグ -xustr=ascii_utf16_ushort を指定するとエラーになります。

このエラーを回避するには、コンパイラが受け入れる C 言語の文法を以前のバージョンの C に変更するオプションを指定する必要があります。使用をお勧めするオプションは -std=c99 または -std=c89 です。

C 言語の文法を変更する次のいずれかのオプションを使用すると、-xustr=ascii_utf16_ushort オプションも受け入れられます: -ansi、-Xc、-Xa、-Xt、または -Xs。

Fortran コンパイラの問題

このリリースの f95 コンパイラでは、次の問題に注意する必要があります。

- 前進なし印刷行の末尾の前にある空白が、出力の位置に影響を与えない。

  出力文の書式の末尾に X 編集記述子を指定しても、出力レコード内の次の文字の位置がその影響を受けません。これによって違いが生まれるのは、出力文に **ADVANCE='NO'** が含まれていて、かつ後続の出力文によって同じレコードに転送される文字がほかに存在している場合です。

  多くの場合、これを回避するには、*n***X** 編集記述子の代わりに空文字列編集記述子を使用します。これらは厳密には同じではありませんが、それは、空文字列編集記述子の場合は実際に空文字がレコード内に挿入されるのに対し、*n***X** では次の *n* 個の文字がスキップされるだけであり、それによって通常は、それらのスキップされた位置がデフォルトで空になるからです。
- 2 つの継続アンパサンドから成る行が存在していると、有効なコードが拒否される。

  単一のアンパサンドを含む空の継続行は、Fortran 規格で禁止されています。しかし、同じ行に 2 つのアンパサンドが含まれていると、規格の制限の対象から外れ、空の継続行を引き続き作成できます。コンパイラではそのような場合は処理されず、エラーが発生します。回

避方法は、その行を削除することであり、それを行っても、プログラムの可読性に影響があるだけであり、意味が付加されることは一切ありません。BOZ 定数が切り捨てられることがある。

- 配列構成のように比較的複雑な一部のシナリオでは、BOZ 定数が、その代入先となると思われる対応する項目が 8 バイト整数エンティティーであっても、デフォルト整数サイズの 4 バイトに切り捨てられる可能性があります。回避方法は、BOZ 定数の代わりに正しい型と種別を持つ定数を、配列構成で使用することです。
- 新しいリリースの丸めアルゴリズムの修正によって、以前のリリースの出力と比べて、`ROUND='NEAREST'` および `ROUCH='COMPATIBLE'` を指定した並び指示、名前並び指示、およびフォーマットされた書き込みの出力に差異が生じることがあります。丸めの差異は、最下位の数字のみである必要があります。

Fortran 77 ライブラリが削除された

ここでは、Oracle Solaris Studio 12.2 リリースで廃止対象の FORTRAN 77 ライブラリが削除されたことに注意してください。これは、レガシー Sun WorkShop `f77` コンパイラでコンパイルされた、共有ライブラリ `libF77`、`libM77`、および `libFposix` に依存する古い実行可能ファイルが動作しないことを意味しています。

配列組み込み関数における大域レジスタの使用

配列組み込み関数の `OT_PRODUCT` および `MATMUL` は、各 SPARC プラットフォームアーキテクチャー用に高度に調整されています。このため、これらの関数は大域レジスタの `%g2`、`%g3`、および `%g4` をスクラッチレジスタとして利用します。

区間演算の場合は、配列組み込みの `ANY`、`ALL`、`COUNT`、`MAXVAL`、`MINVAL`、`SUM`、`PRODUCT` も影響を受けます。

上記の配列組み込み関数を呼び出す場合に、これらのレジスタが一時記憶領域として利用できることを前提にしたユーザーコードを作成しないでください。これらのレジスタ内のデータは、配列組み込み関数を呼び出したときに上書きされます。

アーカイブライブラリ内の F95 モジュールが実行可能ファイルに含まれない

デバッガ `dbx` では、コンパイルに使用されたすべてのオブジェクトファイルが実行可能ファイルの中に含まれている必要があります。通常、ユーザーが追加の作業を実行しなくても、プログラムはこの要件を満たしています。例外となるのは、モジュールを含むアーカイブを使用している場合です。プログラムがモジュールを使用するが、モジュール内の手順または変数をいずれも参照しない場合は、結果として生じるオブジェクトファイルには、モジュール内で定義されるシンボルへの参照は含まれません。オブジェクトファイル内で定義されているシンボルへの参照がある場合のみ、リンカーはアーカイブから取得したオブジェクトファイルをリンクします。その

ような参照がない場合、オブジェクトファイルは実行可能ファイルに含められません。使用されたモジュールに関連付けられているデバッグ情報を見つけようとすると、dbx が警告を出力します。デバッグ情報が見つからないシンボルに関する情報は提供できません。

この問題を回避するためには、-u リンカーオプションを使用します。このオプションは、1 つのシンボルをそのオプション引数として取ります。そのシンボルを未定義のリンカーシンボルのセットに追加し、問題を解決します。モジュールと関連付けられているリンカーシンボルは通常、小文字の文字列に下線が後続するモジュール名です。

たとえば、モジュール MODULE_1 を含むオブジェクトファイルをアーカイブから取り出すには、リンカーオプション -u module_1_ を指定します。f95 コマンドを使用してリンクを実行する場合、コマンド行で -Qoption ld -umodule_1_ を使用してください。

Linux プラットフォームの gethrtime(3F)

システムの省電力が有効になっている場合、AMD プロセッサのクロックレートを正確に取得するための信頼できる方法はありません。結果として、Linux プラットフォーム上で gethrtime(3F) (Fortran コンパイラの、Solaris gethrtime(3C) 関数の Linux 版) に基づく時間関数を使用して実際の時間を高精度で取得する場合、精度の高い値が得られるのは、AMD システムの省電力が無効になっている場合だけです。省電力機能を無効にするには、システムをリブートしなければいけない可能性があります。

ツールの問題

このセクションでは、デバッグツールおよびパフォーマンス分析ツールの既知の制限事項について説明します。

dbx の制限事項と非互換性

dbx には次の制限事項があります。

- libC.so.5 または libC.so.4 の古いコピーを使用すると、C++ 例外の領域で dbx の問題が発生することがあります。不正なスタブや未処理の例外に関する警告メッセージが出力されることがあります。

  回避策: 最新の libC.so.5 をすべてのシステムにインストールしてください。
- Fortran ユーザーは、実行時検査をフル活用できるように、-stackvar オプションを使用してコンパイルするようにしてください。
- 一部のプログラムは、-stackvar オプションを使用すると正しく動作しないことがあります。そのような場合には、RTC を使用せずに配列添字検査を有効にする -C コンパイラオプションを試してください。

- dbx コマンド行インタプリタは、Code Set Independence (CSI) をサポートしない旧バージョンの Korn シェル (ksh) です。マルチバイト文字は、dbx コマンド行に入力すると誤って解釈される場合があります。
- dbx の次の機能は、Linux OS では使用できません。
  - 修正継続
  - マルチスレッドアプリケーションでのパフォーマンスデータの収集。
  - 次のイベントのブレークポイント
    - fault
    - lastrites
    - lwp_exit
    - sysin
    - sysout
    - sync
  - インデックス DWARF (コンパイラオプション -xs=no)
- Linux プラットフォームでのプログラムのデバッグでは、次の問題が発生する可能性があります。
  - 32 ビットプログラムをデバッグするには、-x exec32 オプションを指定して dbx を起動する必要があります。
  - dbx は、exec() の呼び出し時に、Linux プラットフォーム上でフォークされたプロセスを追跡したり新しいプログラムに変更したりできません
  - Linux プラットフォームの場合、Korn シェルの pipe 演算子には制約があります。ターゲットプロセスにアクセスする必要がある dbx コマンドはパイプラインの一部として機能しません。たとえば、次のコマンドではおそらく dbx がハングアップします。

    ```
    where | head -1
    ```

    回避策:
    - Ctrl-C キーを入力して新しい dbx プロンプトを表示します。
    - dbx は多くの情報をキャッシュに格納するので、前述の例の場合、次の一連のコマンドが動作します。

      ```
      where
      where | head –1
      ```

    - リダイレクトコマンドでファイルに出力してから、ファイルの内容を表示します。

      ```
      (dbx) > bag where
      (dbx) cat bag
      ```

- dbx は、GNU C および C++ コンパイラで次の機能をサポートしません。
  - VL 配列 (gcc 4.1 以前)
  - OpenMP
  - RTTI

    - テンプレートの定義
    - デフォルト引数
    - `using` ディレクティブ
    - `friend`
- Oracle Linux 6 の `dbx` には、次の問題があります。
    - システムライブラリ内で間接参照シンボルが使用されていると、`dbx` がブレークポイントを実際の関数にではなくその参照に設定する場合があります。
    - `gcc` 4.4.4 コンパイラを使用してコンパイルされたデバッグコードの場合、`dbx` は `-D` コンパイラオプションを使用して定義されたマクロを認識しないことに注意してください。
- デバッグ情報のスタブ形式は Oracle Solaris Studio でサポートされていますが、Oracle はこの形式をいずれは廃止して、DWARF 形式を優先させる可能性があると公表しています。Oracle は新機能または既存の機能の機能拡張のためにスタブのサポートを実装する必要がありません。

  次のデバッグ機能はスタブ形式でサポートされません。
    - 新しい C++11 の機能。
    - 最適化されたコードのパラメータおよびローカル変数。
    - マクロ (コードが `-g3` オプションを指定してコンパイルされている場合)。
    - デバッグ情報のサブセット (通常、`-xdebuginfo={...}` を指定してコンパイルされている場合)。
- `dbx` がプロセスに接続されているときのデータ収集の問題については、29 ページの「`dbx attach` を使用したプロファイリング (`collect -P`)」を参照してください。

パフォーマンスアナライザおよび `er_print` ユーティリティの制限事項

このセクションでは、パフォーマンスアナライザツールおよび `er_print` ユーティリティの既知の問題について説明します。

- 共有オブジェクトの「ライブラリの可視性」機能が、フィルタリングと組み合わせた場合に必ずしも正しく動作しません。
- パフォーマンスアナライザおよび `er_print` は、jar ファイルに埋め込まれている共有オブジェクトを見つけることができません。また、いくつかの状況で、パフォーマンスアナライザおよび `er_print` が、Java クラスファイルまたはソースファイルを見つけることができません。これらの問題を回避するには、`addpath` を使用して、ファイルが含まれているディレクトリを指します。詳細は、パフォーマンスアナライザのヘルプのトラブルシューティングのセクションを参照してください。
- 実験を比較したときに、いくつかの問題が発生することがあります。
    - フィルタが正しく動作しないことがあります。
    - 「ソース」および「逆アセンブリ」ビューで、包括的メトリックのみが予期されているときに、包括的なメトリックおよび排他的なメトリックが表示されます。

  - 「デュアルソース」ビューのハイパーリンクが動作しないことがあります。
  - 「比較形式」に「デルタ」を設定した場合、デルタ値が「ソース」および「逆アセンブリ」ビューに表示されません。
- CPU の可変クロックレートに関連する次の問題が発生することがあります。
  - 可変クロックレートを持つシステムでは、プロセッサが最大速度より低い速度で実行されてメトリックが回数に変換される場合、サイクルベースのカウンタのハードウェアカウントプロファイリングメトリックが低く報告されます。
  - サイクルベースのカウンタのハードウェアカウンタデータを含む複数の実験を計算または集約するときに、それらの実験が異なるクロックレートを持つマシンで記録された場合、データビューに表示される時間メトリックが正確なのは最初の実験のみです。ただし、「概要」に表示されるメトリックは正確です。
  - 異なるクロック周波数で動作する複数の CPU を持つシステムの実験は、1 つの CPU のクロックレートに基づいて進められるため、ほかの CPU 上のイベントが過大または過小にカウントされる可能性があります。
- SPARC T5、M5、および M6 システムで多数のスレッドを使用するプロファイリング OpenMP アプリケーションが、読み込むことができない非常に大きな実験を生成することがあります。回避策は、`collect` を実行する前に、環境変数 `SP_COLLECTOR_NO_OMP` を設定することです。この変数を設定すると、結果の実験の OpenMP 構文およびメトリックが表示されず、「ユーザー」、「上級」、および「マシン」ビューモードがすべて同じように表示されます。
- パフォーマンスアナライザは、プロファイリング中に呼び出しスタックが記録されなかった実験を破棄します。
- ホスト名が `/etc/hosts` にない場合、パフォーマンスアナライザを開始できません。回避策は、ホスト名が `/etc/hosts` にあることを確認することです。

リモートホスト上のプロファイリングアプリケーションの制限事項

パスフレーズを使用したリモートログインはまだサポートされていません。

`collect` ユーティリティ

このセクションでは、`collect` ユーティリティー、およびパフォーマンスアナライザツールのデータ収集の既知の問題について説明します。

- JDK 1.7.0_40 から JDK 1.7.0_59 を使用した場合、JDK のバグが原因で、Java および C++ のアプリケーションが混在したプロファイリングが完了しないことがあります。Java から C++ への呼び出しのスタックが正しく巻き戻されません。
- Linux のヒープトレースで `calloc` への呼び出しがトレースされません。
- いくつかの状況では、ハードウェアカウンタプロファイリングに `~system=1` 属性を使用すると、アプリケーションで障害が発生するか、Oracle Solaris オペレーティングシステムがクラッシュまたはリセットされることがあります。

- -std=C++11 を指定してコンパイルされたバイナリに対して、カウントデータ (collect -c) の収集が動作しません。

er_kernel ユーティリティー

このセクションでは、er_kernel ユーティリティーの既知の問題について説明します。

- er_kernel -t が要求された間隔より数秒間長く実行されることがあります。
- リブートを誘発する Oracle VM のバグを回避するために、Oracle VM の配下で実行されている Oracle Solaris システムでは、er_kernel が実行されません。
- DTrace スタック巻き戻しで 1 つのフレームが省略されることがありますが、それは特に、リーフ関数がそのエピローグ内にある場合に発生します。

dbx collector を使用したプロファイリング

このセクションでは、dbx collector コマンドを使用したときにアプリケーションのプロファイリングで発生する問題について説明します。

- Java で dbx collector を使用する場合の制限事項があります。dbx ターゲットに Java クラスファイルを指定できません。代わりに、JVM をターゲットとして指定し、クラスファイルを dbx run コマンドのパラメータとして指定する必要があります。例:

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

- Linux 上で、dbx または collect -P を使用した Java アプリケーションのプロファイリングが失敗することがあります。
- dbx でハードウェアカウンタプロファイリングを実行する場合は、hi または lo を指定した collector hwprofile counter または collector hwprofile addcounter コマンドを発行する前に、collector hwprofile on コマンドを発行する必要があります。そうしないと、エラーが報告されます。

  例:

  ```
  > 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 を使用したプロファイリング (collect -P)

このセクションでは、dbx attach コマンドを使用したときに、実行中のアプリケーションのプロファイリングで発生する問題について説明します。

- コレクタライブラリ libcollector.so を事前にロードせずに、LD_PRELOAD なしで開始された実行中のプロセスに dbx を接続すると、さまざまなエラーが発生する可能性があります。どのトレーシングデータも収集できません。同期はトレーシング、ヒープトレーシング、入出力トレーシング、または MPI トレーシングを待機します。トレースデータの収集は各種ライブラリに割り込むことで行われますが、libcollector.so が事前にロードされていなければ、割り込みを行えません。
- dbx がプロセスに接続されたあとにターゲットプログラムがシグナルハンドラをインストールし、そのシグナルハンドラが Oracle Solaris の SIGPROF シグナルおよび SIGEMT シグナル、または Linux の SIGIO シグナルを転送しない場合、すべてまたは一部のプロファイリングデータが失われることがあります。
- ターゲットプログラムが libcpc.so を使用している場合、コレクタとプログラムが両方ともこのライブラリを使用するため、ハードウェアカウンタ実験が失敗することがあります。
- ターゲットプログラムが setitimer(2) を呼び出す場合、コレクタとプログラムの両方がタイマーを使用しているため、クロックプロファイリング実験に失敗する場合があります。
- malloc ライブラリに呼び出しを行なっているターゲットに接続すると、ターゲットで障害が発生することがあります。接続しているときにハードウェアカウンタを要求すると、そのような障害が発生する可能性が非常に高くなります。
- collector java on コマンドまたは collect -j on -P *pid* を使用しない場合、Java プログラムへの接続が失敗します。
- Linux では、dbx または collect -P を使用した Java プログラムへの接続が失敗することがあります。
- Linux では、ブロッキング呼び出しまたは非ブロッキング呼び出しを行なっているターゲットに接続すると、ターゲットで障害が発生することがあります。
- Linux では、マルチスレッドのターゲットに接続すると、接続の時点ですでに作成されているスレッドのデータが正しく記録されません。欠落しているデータに関する警告が表示されません。JVM はマルチスレッドであるため、これには Java ターゲットが含まれます。

IDE の制限事項

このセクションでは、IDE の既知の制限事項について説明します。

リモートホストでバイナリからプロジェクトを作成するには、リモートホストで Oracle Solaris Studio 12.4 または Oracle Solaris Studio 12.3 が利用可能である必要があります。この機能は、以前のリリースではサポートされません。

コード分析ツールの制限事項

このセクションでは、コードアナライザツールの既知の制限事項について説明します。

- プログラムのサイズ (テキスト + データ) が `-xmodel=small` (2^{32} - 2^{24} - 1) の最大範囲を超えるほどに大きい場合、`discover` および `uncover` ツールは `-xmodel=medium` を指定してコンパイルされた x86_64 バイナリに対して動作しません。
- `discover` ツールは、Oracle Solaris 11.2 でリンカーオプション `-z ancillary` を指定して作成されたデバッグ情報を含むファイルである付属ファイルに対して動作しません。
- `discover` ツールは、C++11 を指定してコンパイルされた UMR および PIR を報告しません。
- Oracle Linux でコード分析ツールを使用するには、元のバイナリを `-xannotate=[yes]` オプションを指定してコンパイルおよびリンクする必要があります。