インテル® C++ Composer XE 2011 Linux* 版インストール・ガイドおよびリリースノート

資料番号: 321412-003JA

2012 年 4 月 3 日

目次

1 概要

このドキュメントでは、製品のインストール方法、新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題について説明します。

インテル® C++ Composer XE 2011 は、以前「インテル® C++ コンパイラー・プロフェッショナル・エディション」と呼ばれていた製品の最新バージョンです。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。

Update 10 (2011.10)

- インテル® C++ コンパイラー XE 12.1.4
- インテル® デバッガー 12.1.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 7
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 4
- 報告された問題の修正

Update 9 (2011.9)

- インテル® C++ コンパイラー XE 12.1.3
- インテル® デバッガー 12.1.3
- インテル® マス・カーネル・ライブラリー 10.3 Update 9
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 3
- 報告された問題の修正

Update 8 (2011.8)

- インテル® C++ コンパイラー XE 12.1.2
- インテル® デバッガー 12.1.2
- インテル® マス・カーネル・ライブラリー 10.3 Update 8
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 6
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 2
- Pardus* 2011.2 のサポート (x64 のみ)
- 報告された問題の修正

Update 7 (2011.7)

- インテル® C++ コンパイラー XE 12.1.1
- インテル® デバッガー 12.1.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 7
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 1
- インライン・アセンブリーと組込み関数でのインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX 2) のサポート
- 報告された問題の修正

Update 6 (2011.6)

- 新しいトップレベル・フォルダーへの製品のインストール
- クラスターでのインストールのサポート
- インテル® C++ コンパイラー XE 12.1
  - 追加の C++0x 機能のサポート
  - コンパイラー・オプションの追加
  - インテル® Cilk™ Plus サポートの拡張
  - インテル® Cilk™ Plus の配列表記 (アレイ・ノーテーション) セマンティクスの変更
  - OpenMP* サポートの拡張
  - コンパイラーの主要ドキュメントであるユーザー・リファレンス・ガイドの簡略化と再編成。主な変更点: インテル® コンパイラーの主要機能をまとめた新しいセクション「主な機能」の追加と、コンパイラー・オプション・リファレンスの機能別のグループ化。
- インテル® デバッガー 12.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 6
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 5
- インテル® スレッディング・ビルディング・ブロック 4.0
- Eclipse* 3.7 + CDT 8.0 への統合のサポート
- 報告された問題の修正

Update 5 (2011.5)

- インテル® マス・カーネル・ライブラリー 10.3 Update 5
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 8
- 報告された問題の修正

Update 4 (2011.4)

- インテル® マス・カーネル・ライブラリー 10.3 Update 4
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 4
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 7
- `-export` および `-export-dir` のサポート終了予定
- 報告された問題の修正

Update 3 (2011.3)

- インテル® マス・カーネル・ライブラリー 10.3 Update 3
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 3
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 6
- `-sox` オプションの拡張
- 報告された問題の修正

Update 2 (2011.2)

- インテル® マス・カーネル・ライブラリー 10.3 Update 2
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 2
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 5
- immintrin.h の 3 つの組込み関数の変更
- “inspxe-runsc” ユーティリティーの変更
- 報告された問題の修正

Update 1 (2011.1)

- インテル® マス・カーネル・ライブラリー 10.3 Update 1
- 報告された問題の修正

製品リリース (2011.0)

- 最初の製品リリース

1.2 製品の内容

インテル® C++ Composer XE 2011 Update 10 Linux* 版には、次のコンポーネントが含まれています。

- インテル® C++ コンパイラー XE 12.1.4。Linux* オペレーティング・システムを実行する IA-32 およびインテル® 64 アーキテクチャー・システムで動作するアプリケーションをビルドします。
- インテル® デバッガー 12.1.4
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 7
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 4
- Eclipse* 開発環境への統合
- 各種ドキュメント

1.3 動作環境

アーキテクチャー名についての説明は、http://intel.ly/q9JVjE (英語) を参照してください。

- インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) 対応の IA-32 またはインテル® 64 アーキテクチャー・プロセッサーをベースとするコンピューター (インテル® Pentium® 4 プロセッサー以降、または互換性のあるインテル以外のプロセッサー)
    - 64 ビット・アプリケーションの開発は、64 ビット・バージョンの OS でのみサポートしています。32 ビット・アプリケーションの開発は、32 ビット・バージョンまたは 64 ビット・バージョンの OS のいずれかでサポートしています。
    - 64 ビット・バージョンの OS で 32 ビット・アプリケーションを開発する場合は、Linux* ディストリビューションからオプションのライブラリー・コンポーネント (ia32-libs、lib32gcc1、lib32stdc++6、libc6-dev-i386、gcc-multilib) をインストールする必要があります。
- 機能を最大限に活用できるよう、マルチコアまたはマルチプロセッサー・システムの使用を推奨します。
- RAM 1GB (2GB 推奨)
- 2GB のディスク空き容量 (すべての機能をインストールする場合)
- 次の Linux* ディストリビューションのいずれか (本リストは、インテル社により動作確認が行われたディストリビューションのリストです。その他のディストリビューションでも動作する可能性はありますが、推奨しません。ご質問は、テクニカルサポートまでお問い合わせください。)
    - Asianux* 3.0、4.0 (サポート終了予定)
    - Fedora* 15
    - Red Hat* Enterprise Linux* 4 (サポート終了予定)、5、6
    - SUSE LINUX Enterprise Server* 10、11 SP1
    - Ubuntu* 10.04 LTS、11.04
    - Debian* 6.0
    - Pardus* 2011.2 (x64 のみ)
- Linux* 開発ツール・コンポーネント (gcc、g++ および関連ツールを含む)
- `-traceback` オプションを使用するには、libunwind.so が必要です。一部の Linux* ディストリビューションでは、別途入手して、インストールする必要があります。

インテル® デバッガーのグラフィカル・ユーザー・インターフェイスを使用するためのその他の要件

- Java* ランタイム環境 (JRE) 5.0 (1.5) または 6.0 (1.6) – 5.0 推奨
    - IA-32 アーキテクチャー・システムでは 32 ビット版の JRE、インテル® 64 アーキテクチャー・システムでは 64 ビット版の JRE を使用する必要があります。

Eclipse 開発環境に統合するためのその他の条件*

- 次のいずれかが必要です。
    - Eclipse* Platform 3.6 および次の両方
        - Eclipse* C/C++ Development Tools (CDT) 7.0 以降
        - Java* ランタイム環境 (JRE) 5.0 (1.5) 以降
    - Eclipse* Platform 3.7 および次の両方
        - Eclipse* C/C++ Development Tools (CDT) 8.0 以降
        - Java* ランタイム環境 (JRE) 6.0 (1.6) 以降

† JRE 6.0 の Update 10 以前には、インテル® 64 アーキテクチャーでクラッシュするという既知の問題があります。JRE の最新のアップデートを使用することを推奨します。詳細は、http://www.eclipse.org/eclipse/development/readme_eclipse_3.7.html を参照してください。

説明

- インテル® コンパイラーは、さまざまな Linux* ディストリビューションと gcc バージョンで動作確認されています。一部の Linux* ディストリビューションには、動作確認されたヘッダーファイルとは異なるバージョンのものが含まれており、問題を引き起こすことがあります。使用する glibc のバージョンは、gcc のバージョンと同じでなければなりません。最良の結果を得るため、上記のディストリビューションで提供されている gcc バージョンのみを使用してください。
- インテル® コンパイラーは、デフォルトで、インテル® SSE2 命令対応のプロセッサー (例: インテル® Pentium® 4 プロセッサー) が必要な IA-32 アーキテクチャー・アプリケーションをビルドします。コンパイラー・オプションを使用して任意の IA-32 アーキテクチャー・プロセッサー上で動作するコードを生成できます。ただし、アプリケーションでインテル® インテグレーテッド・パフォーマンス・プリミティブまたはインテル® スレッディング・ビルディング・ブロックを使用している場合、そのアプリケーションの実行には、インテル® SSE2 命令対応のプロセッサーが必要です。
- 非常に大きなソースファイル (数千行以上) を -O3、-ipo および -openmp などの高度な最適化オプションを使用してコンパイルする場合は、多量の RAM が必要になります。
- 上記のリストにはすべてのプロセッサー・モデル名は含まれていません。リストされているプロセッサーと同じ命令セットを正しくサポートしているプロセッサー・モデルでも動作します。特定のプロセッサー・モデルについては、テクニカルサポートにお問い合わせください。
- 一部の最適化オプションには、アプリケーションを実行するプロセッサーの種類に関する制限があります。詳細は、オプションの説明を参照してください。

1.3.1 Red Hat* Enterprise Linux* 4 のサポート終了予定

インテル® Composer XE の将来のメジャーリリースでは、Red Hat* Enterprise Linux* 4 はサポートされなくなる予定です。これらのオペレーティング・システムを使用している場合は、インテルでは新しいバージョンへの移行を推奨しています。

1.3.2 Asianux* のサポート終了予定

インテル® C++ Composer XE の将来のメジャーリリースでは、Asianux* のすべてのディストリビューションはサポートされなくなる予定です。

1.3.3 IA-64 アーキテクチャー (インテル® Itanium®) 開発の未サポート

本バージョンでは、IA-64 アーキテクチャー (インテル® Itanium®) システム上、または IA-64 アーキテクチャー・システム向けの開発をサポートしていません。インテル® コンパイラー 11.1 ではまだサポートされています。

1.4 ドキュメント

製品ドキュメントは、「インストール先フォルダー」で示されているように、Documentation フォルダーに保存されています。

最適化に関する注意事項

インテル® コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

改訂 #20110804

1.5 日本語サポート

インテル® コンパイラーは、日本語ユーザー向けのサポートを提供しています。エラーメッセージ、ビジュアル開発環境ダイアログ、ドキュメントの一部が英語のほかに日本語でも提供されています。エラーメッセージやダイアログの言語は、システムの言語設定に依存します。日本語版ドキュメントは、Documentation および Samples ディレクトリー以下の ja_JP サブディレクトリーにあります。

日本語サポートはすべての製品アップデートで提供されているわけではありません。

日本語サポート版を英語のオペレーティング・システムで使用する場合や日本語のオペレーティング・システムで英語サポート版を使用する場合は、http://intel.ly/qhINDv (英語) の説明を参照してください。

1.6 テクニカルサポート

インテル® ソフトウェア開発製品レジストレーション・センターでラインセンスを登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

テクニカルサポート、製品のアップデート、ユーザーフォーラム、FAQ、ヒント、およびその他のサポート情報は、http://www.intel.com/software/products/support/ (英語) を参照してください。

注: 代理店がテクニカルサポートを提供している場合は、インテルではなく代理店にお問い合わせください。

2 インストール

本製品のインストールには、有効なライセンスファイルまたはシリアル番号が必要です。本製品を評価する場合には、インストール時に [製品を評価する (シリアル番号不要)] オプションを選択してください。

DVD 版を購入した場合は、DVD をドライブに挿入し、DVD のトップレベル・ディレクトリーにディレクトリーを変更 (cd) して、次のコマンドでインストールを開始します。

```
./install.sh
```

ダウンロード版を購入した場合は、次のコマンドを使用して、書き込み可能な任意のディレクトリーに展開します。

```
tar –xzvf name-of-downloaded-file
```

その後、展開したファイルを含むディレクトリーに移動 (cd) し、次のコマンドでインストールを開始します。

```
./install.sh
```

手順に従ってインストールを完了します。

利用可能なダウンロード・ファイルには各種あり、それぞれ異なるコンポーネントの組み合わせを提供していることに注意してください。ダウンロード・ページを注意深くお読みになり、適切なファイルを選択してください。

新しいバージョンをインストールする前に古いバージョンをアンインストールする必要はありません。新しいバージョンは古いバージョンと共存可能です。

2.1.1 インテル® C++ コンパイラー 11.1 プロフェッショナル・エディションのライセンスまたはシリアル番号によるインストール

インテル® C++ コンパイラー 11.1 プロフェッショナル・エディションのライセンスおよびシリアル番号は、インテル® C++ Composer XE 2011 では使用できません。製品のサポートステータスがアクティブな場合、以下の手順に従って、新しいアップグレード・ライセンスおよびシリアル番号を無料で取得できます。

1. インテル® ソフトウェア開発製品レジストレーション・センターの [登録ユーザーのログイン] セクションで、[ログイン ID] と [パスワード] を入力してサイトにログインします。サポートサービスが有効なすべての製品のリストが表示されます。
2. 旧名称の製品とともに、XE 製品の名前も表示されます。[最新版アップデートのダウンロード] 列で最新のアップデートのリンクをクリックすると、製品のアップグレード・ページが表示されます。アップグレードする製品名をクリックしてください。
3. アップデートされたライセンスファイルを登録アドレスへメールで送信するか、表示されるシリアル番号を使用して、インテル® C++ Composer XE 2011 製品をインストールすることができます。

2.1.2 クラスターでのインストール (Update 6 以降)

インストールするマシンにインテル® Cluster Studio XE のライセンスがあり、クラスターメンバーの場合、「フル・インストール」を選択すると、そのクラスターのアクセス可能なすべてのノードに製品がインストールされます。「カスタム・インストール」を選択すると、現在のノードのみにインストールするオプションを選択できます。

2.1.3 インテルのアクティベーション・ツールを使用した製品のアクティベーション

この製品リリースでは、新しいインテルのアクティベーション・ツール "Activate" が /opt/intel/ActivationTool/Activation/ ディレクトリーにインストールされます。

インストール中に評価用ライセンスまたは評価用シリアル番号を使用したり、あるいは [製品を評価する (シリアル番号不要)] オプションを選択して製品をインストールした場合、製品を購入した後にこのアクティベーション・ツール (/opt/intel/ActivationTool/Activation/Activate) を使用して製品をアクティベートできます。これにより、評価版から製品版へ移行することができます。このツールを使用するには、次のコマンドを実行します。

```
$ /opt/intel/ActivationTool/Activation/Activate [シリアル番号]
```

2.1.4 サイレントインストール

自動インストール、「サイレント」インストール機能についての詳細は、http://intel.ly/ngVHY8 (英語) を参照してください。

2.1.5 ライセンスサーバーの使用

「フローティング・ライセンス」を購入された場合は、ライセンスファイルまたはライセンスサーバーを使用したインストール方法について http://intel.ly/pjGfwC (英語) を参照してください。この記事には、多様なシステムにインストールすることができるインテル・ライセンス・サーバーに関する情報も記述されています。

2.1.6 Eclipse* 統合のインストール

「Eclipse* 統合」セクションを参照してください。

2.1.7 既知のインストールの問題

- Linux* ディストリビューションの Security-Enhanced Linux (SELinux) 機能を有効にしている場合は、インテル® C++ コンパイラーをインストールする前に SELINUX モードを permissive に変更する必要があります。詳細は、Linux* ディストリビューションのドキュメントを参照してください。インストールが完了したら、SELINUX モードを元の値に戻してください。
- 一部の Linux* バージョンでは、自動マウントデバイスに "実行" 許可がなく、インストール・スクリプトを直接 DVD から実行すると、次のようなエラーメッセージが表示されることがあります。

  ```
  bash:./install.sh:/bin/bash: bad interpreter:Permission denied
  ```

  このエラーが表示された場合は、次の例のように実行許可を含めて DVD を再マウントします。

  ```
  mount /media/<dvd_label> -o remount,exec
  ```

  その後、再度インストールを行ってください。
- 「システム要件」に記述されているように、本バージョンでは、IA-32 およびインテル® 64 アーキテクチャー・ベースのシステムで Debian* または Ubuntu* をサポートしています。ただし、ライセンス・ソフトウェアの制約上、Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システム上では、インストール時に [製品を評価する (シリアル番号不要)] オプションで IA-32 コンポーネントをインストールできません。これは、[製品を評価する (シリアル番号不要)] オプションを使用する場合のみの問題です。シリアル番号、ライセンスファイル、フローティング・ライセンス、その他のライセンス・マネージャー操作、およびオフラインでのアクティベーション操作 (シリアル番号

を使用) には、影響はありません。Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システムで、本バージョンの IA-32 コンポーネントの評価が必要な場合は、インテル® ソフトウェア評価センター (http://intel.ly/nJS8y8 (英語)) で評価版のシリアル番号を入手してください。

2.2 インストール先フォルダー

コンパイラーは、デフォルトでは /opt/intel にインストールされます。本リリースノートでは、この場所を <install-dir> と表記します。コンパイラーは、別の場所にインストールしたり、“非 root” で任意の場所にインストールすることもできます。

本リリースではディレクトリー構成が インテル® コンパイラー 11.1 から変更されています。

インテル® C++ Composer XE 2011 の以前のリリースと Update 6 では、トップレベルのインストール・ディレクトリーが異なりますが、引き続き composerxe シンボリック・リンクを使用して最新の製品インストールを参照することができます。

<install-dir> 以下には次のサブディレクトリーがあります。

- bin - インストールされている最新バージョンの実行ファイルへのシンボリック・リンク
- lib - インストールされている最新バージョンの lib ディレクトリーへのシンボリック・リンク
- include - インストールされている最新バージョンの include ディレクトリーへのシンボリック・リンク
- man - インストールされている最新バージョンの man ページが含まれているディレクトリーへのシンボリック・リンク
- ipp - インストールされている最新バージョンのインテル® インテグレーテッド・パフォーマンス・プリミティブのディレクトリーへのシンボリック・リンク
- mkl - インストールされている最新バージョンのインテル® マス・カーネル・ライブラリーのディレクトリーへのシンボリック・リンク
- tbb - インストールされている最新バージョンのインテル® スレッディング・ビルディング・ブロックのディレクトリーへのシンボリック・リンク
- composerxe – composer_xe_2011_sp1 ディレクトリーへのシンボリック・リンク
- composer_xe_2011_sp1 - インストールされている最新バージョンのインテル® Composer XE 2011 コンパイラーのサブディレクトリーへのシンボリック・リンク
- composer_xe_2011_sp1.<n>.<pkg> - 特定のリビジョン番号のファイルが含まれている物理ディレクトリー。<n> はリビジョン番号、<pkg> はパッケージビルド ID。

各 composer_xe_2011_sp1 ディレクトリーには、インストールされている最新のインテル® Composer XE 2011 コンパイラーを参照する次のサブディレクトリーが含まれています。

- bin - コンパイラー環境とホスト環境用のコンパイラー実行ファイルへのシンボリック・リンクを設定するためのスクリプト
- pkg_bin - コンパイラーの bin ディレクトリーへのシンボリック・リンク
- include - コンパイラーの include ディレクトリーへのシンボリック・リンク
- lib - コンパイラーの lib ディレクトリーへのシンボリック・リンク
- ipp - ipp ディレクトリーへのシンボリック・リンク
- mkl - mkl ディレクトリーへのシンボリック・リンク
- tbb - tbb ディレクトリーへのシンボリック・リンク
- debugger – debugger ディレクトリーへのシンボリック・リンク
- eclipse_support – eclipse_support ディレクトリーへのシンボリック・リンク
- man - man ディレクトリーへのシンボリック・リンク
- Documentation – Documentation ディレクトリーへのシンボリック・リンク
- Samples – Samples ディレクトリーへのシンボリック・リンク

各 composer_xe_2011_sp1.<n>.<pkg> ディレクトリーには、特定のリビジョン番号のインテル® Composer XE 2011 コンパイラーを参照する次のサブディレクトリーが含まれています。

- bin - すべての実行ファイル
- compiler - 共有ライブラリーとヘッダーファイル
- debugger – デバッガーファイル
- Documentation – ドキュメント・ファイル
- man – man ページ
- eclipse_support – Eclipse 統合をサポートするためのファイル
- ipp - インテル® インテグレーテッド・パフォーマンス・プリミティブのライブラリーとヘッダーファイル
- mkl - インテル® マス・カーネル・ライブラリーのライブラリーとヘッダーファイル
- tbb - インテル® スレッディング・ビルディング・ブロックのライブラリーとヘッダーファイル
- Samples - サンプルプログラムとチュートリアル・ファイル

インテル® C++ コンパイラーとインテル® Fortran コンパイラーの両方がインストールされている場合、所定のバージョンおよびリビジョン番号のフォルダーが共有されます。

このディレクトリー構成により、任意のバージョン/リビジョン番号のインテル® Composer XE 2011 コンパイラーを選択することができます。<install-dir>/bin にある compilervars.sh [.csh] スクリプトを参照すると、インストールされている最新のコンパイラーが使用されます。このディレクトリー構成は、将来のリリースでも保持される予定です。

2.3 削除/アンインストール

製品の削除 (アンインストール) は、製品をインストールしたユーザー (root または非 root ユーザー) で実行してください。インストールに sudo を使用した場合は、アンインストールの際にも使用する必要があります。インストールされているパフォーマンス・ライブラリー・コンポーネントや Eclipse* 統合コンポーネントを残したまま、コンパイラーのみを削除することはできません。

1. 端末を開いて、<install-dir> 以外のフォルダーに移動 (cd) します。
2. その後、次のコマンドを使用します。<install-dir>/bin/uninstall.sh
3. 画面の指示に従ってオプションを選択します。
4. 別のコンポーネントを削除するには、ステップ 2 と 3 を繰り返します。

同じバージョンのインテル® Fortran コンパイラーをインストールしている場合は、Fortran コンパイラーも削除されます。

使用している Eclipse* にインテル® C++ コンパイラーの Eclipse* 統合機能が追加されている場合は、Eclipse* の構成からインテルの統合拡張を削除して、構成を更新する必要があります。そのためには、[Help (ヘルプ)] メニューから [About Eclipse (Eclipse について)] を開いて [Installation Details (インストール詳細)] をクリックします。そして、[Installed Software (インストール済みのソフトウェア)] から [Intel(R) C++ Compiler XE 12.1 for Linux* OS (インテル (R) C++ Compiler XE 12.0 Linux* OS 版)]" を選択して [Uninstall... (アンインストール...)] をクリックします。処理が完了したら [Finish (完了)] をクリックして、Eclipse* の再起動を求められたら [Yes (はい)] を選択します。

3 インテル® C++ コンパイラー

このセクションでは、インテル® C++ コンパイラーの変更点、新機能、および最新情報をまとめています。

3.1 互換性

バージョン 11.0 では、IA-32 システムのデフォルトでのコード生成において、アプリケーションを実行するシステムでインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) がサポートされていると仮定するように変更されました。詳細は、下記を参照してください。

3.2 新機能と変更された機能

インテル® C++ Composer XE 2011 Update 7 には、インテル® C++ コンパイラー XE 12.1 が含まれています。このバージョンでは、次の機能が新たに追加または大幅に拡張されています。これらの機能に関する詳細は、ドキュメントを参照してください。

- C++0x からの機能
  - 可変個引数テンプレート
  - char16_t/char32_t 型
  - 関数の default/delete 指定
  - sizeof、typeid 、decltype によるクラスの非スタティック・データ・メンバーの直接参照
  - ヌルポインター
  - N2634 で説明されている新しい SFINAE
  - テンプレート・エイリアス
  - 後置する戻り値
  - 関数テンプレート用のデフォルトのテンプレート引数
- OpenMP* 3.1
  - final 節
  - mergeable 節
  - taskyield 宣言子
  - 新しい atomic 節
- スタティック・セキュリティー解析の IDE 統合の向上
- インテル® Cilk™ Plus の拡張:
  - 新しい holder ハイパーオブジェクト
  - simd プラグマのループと要素関数の制限が緩和。入れ子したループ、配列表記 (アレイ・ノーテーション) 、switch 文、break/continue 文を含めることができます。
  - 新しい “ストライド 0” 配列表記による多次元部分配列での複数の次元にわたる複製
  - 異なるアーキテクチャー向けによりスケーラブルなベクトル長の指定が可能な新しい vectorlengthfor 節を simd プラグマと要素関数に追加

インテル® C++ コンパイラー XE 12.0 では、次の機能が新たに追加または大幅に拡張されています。これらの機能に関する詳細は、ドキュメントを参照してください。

- インテル® Cilk™ Plus。インテル® C++ コンパイラー向けのこの言語拡張を使用することで、新規および既存のソフトウェアを簡単に並列化できます。
- ガイド付き自動並列化
- C++0x からの機能
  - 右辺値参照
  - 標準的なアトミック演算
  - “Windows* C++” モードでの C99 の 16 進浮動小数点定数のサポート
  - 右角括弧
  - 拡張 friend 宣言
  - 混在した文字列リテラルの結合
  - `long long` のサポート
  - 可変引数マクロ
  - スタティック・アサーション

    - auto 型変数
    - extern テンプレート
    - __func__ 事前定義済み指定子
    - 式の型宣言 (decltype)
    - ユニバーサル文字名
    - 強い型付けの列挙型
    - ラムダ
- より高速でやや精度が低い算術ライブラリー関数を使用するためのオプション
- プロセッサーのモデルや製造元に関係なく一貫した結果を返す算術ライブラリー関数を使用するためのオプション

3.2.1 インテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX 2) 命令のサポート (Update 7)

インテル® C++ Composer XE 2011 Update 7 のコンパイラーは、インライン・アセンブリーと 組込み関数 (immintrin.h) でインテル® AVX 2 命令をサポートします。

3.2.2 インテル® Cilk™ Plus の配列表記 (アレイ・ノーテーション) セマンティクスの変更 (Update 6)

インテル® C++ Composer XE 2011 では、次のようなインテル® Cilk™ Plus の部分配列の代入は、結果の一時コピーが生成されるためパフォーマンスに影響します。

```
a[:]= b[:]+ c[:];
```

インテル® C++ Composer XE 2011 Update 6 から、代入式の右辺の部分配列 (上記の例では b[:] や c[:]) の一部が左辺の部分配列 (上記の例では a[:]) とメモリー上でオーバーラップする場合、そのような代入の結果は不定となります。意図する動作が得られるように、代入式でメモリー上の部分的なオーバーラップが発生しないようにするのはプログラマーの責任です。

ただし、次のように、部分配列が完全にオーバーラップする場合は例外です。

```
a[:]= a[:]+ 3;
```

この場合、配列が完全にオーバーラップするため、意図したとおりに動作し、期待どおりの結果が得られます。

3.2.3 インテル® Cilk™ Plus の “scalar” 節のサポート終了予定

インテル® Cilk™ Plus の要素関数のオプションで使用可能な “scalar” 節は、将来のバージョンのインテル® C++ Composer XE では削除されます。代わりに、機能的に同じ “uniform” 節を使用してください。

3.2.4 -export および -export-dir のサポート終了予定 (Update 4)

2 つのコンパイラー・オプション -export と -export-dir は、C++ テンプレートのエクスポート機能をサポートするためのものです。この機能は、C++0x 標準でサポートされる予定でしたが、サポートされなくなりました。インテル® コンパイラーでは、この機能のサポートを終了し、将来のリリースでは削除される予定です。

3.2.5 -sox オプションの追加キーワード、デフォルトの変更 (Update 3)

オブジェクト・ファイルおよび実行ファイルに使用されたコンパイラー・オプションとプロシージャーのプロファイル情報を追加するための -sox オプションは、インライン展開された関数のリストを含めたり、プロシージャーのプロファイル情報を除外したり指定できるようになりました。

-sox の構文は次のように変更されました。

```
-[no]sox
-sox=keyword[,keyword]
```

keyword には、inline または profile のいずれかを指定できます。キーワードなしで -sox を指定すると、以前のバージョンとは異なり、コマンドライン・オプションの情報のみが追加されます。以前のリリースと同じ動作にするには、-sox=profile を使用してください。-sox オプションはコマンドラインで複数回指定することができますが、その場合は左から右の順に解釈されます。

3.2.6 3 つの組込み関数の変更 (Update 2)

3 つの組込み関数 (_rdrand16_step()、_rdrand32_step()、_rdrand64_step()) は、Update 2 で変更されました。この変更は、ドキュメントにはまだ反映されていません。これらの組込み関数は、"immintrin.h" ヘッダーファイルで宣言されており、ハードウェアにより生成される乱数値を返します。

これらの組込み関数は、1 つの RDRAND 命令にマップされ、16/32/64 ビットの整数型の乱数を生成します。

構文

1. extern int _rdrand16_step(unsigned short *random_val);
2. extern int _rdrand32_step(unsigned int *random_val);
3. extern int _rdrand64_step(unsigned __int64 *random_val);

説明

これらの組込み関数は、RDRAND 命令を使用して、ハードウェアにより生成される乱数値の取得を 1 回試みます。生成された乱数値は指定されたメモリー位置に書き込まれ、成功ステータスが返されます。ハードウェアにより有効な乱数値が返された場合は 1 を返し、そうでない場合は 0 を返します。

戻り値

ハードウェアにより生成された 16/32/64 乱数値。

制限事項

_rdrand64_step() 組込み関数は、64 ビット・レジスター対応のシステムでのみ使用できます。

3.2.7 スタティック解析機能 (旧:「スタティック・セキュリティー解析」または「ソースチェッカー」) にはインテル® Inspector XE が必要

バージョン 11.1 の「ソースチェッカー」機能が拡張され、「スタティック解析」に名称が変更されました。スタティック解析を有効にするコンパイラー・オプションはバージョン 11.1 と同じですが (例: -diag-enable sc)、解析結果がコンパイラー診断結果ではなく、インテル® Inspector XE で表示可能なファイルに出力されるようになりました。

3.2.7.1 Update 2 からの "inspxe-runsc" コマンドライン・ユーティリティーの変更

インテル® Composer XE 2011 に含まれるこのユーティリティーは、Update 2 から変更されています。この変更は、インテル® Composer XE 2011 を使用してスタティック解析を実行する場合にのみ影響します。スタティック解析を使用しない場合や、このユーティリティーを使用せずにスタティック解析を実行する場合には影響ありません。スタティック解析はインテル® Parallel Studio XE 2011、インテル® Fortran Studio XE 2011 またはインテル® C++ Studio XE 2011 でのみ利用できます。そのため、これらの製品をお使いでない場合は影響ありません。

inspxe-runsc は、アプリケーションのビルド方法を示す**ビルド仕様**を実行します。通常、ビルド仕様ファイルは、ビルドを実行して、実際に行われたコンパイルとリンクを記録することにより生成されます。inspxe-runsc は、インテル® コンパイラーをスタティック解析モードで使用して、再度この処理を行います。スタティック解析結果はリンクステップで生成されるため、inspxe-runsc で複数のリンクステップを持つビルドが含まれるビルド仕様を実行すると、複数のスタティック解析結果が生成されます。

インテル® Composer XE 2011 およびインテル® Composer XE 2011 Update 1 の inspxe-runsc は、すべてのスタティック解析結果を同じディレクトリーに生成します。リンクが複数ある場合、これは、1 つのプロジェクトのスタティック解析結果は同じディレクトリーに 1 つだけでなければならないという規則に違反します。新しいバージョンの inspxe-runsc は、リンクステップごとの結果を個別のディレクトリーに生成することで、この規則に従っています。ディレクトリー名は、リンクされるファイルの名前を基に付けられます。2 つの実行ファイル file1.exe と file2.exe をビルドするプロジェクトのビルド仕様の場合、以前のバージョンの inspxe-runsc では、file1 の結果と file2 の結果 (例えば r000sc と r001sc) が同じディレクトリーに作成されます。新しいバージョンの inspxe-runsc でも結果は 2 つ作成されますが、file1 の結果は “My Inspector XE results – file1/r000sc”、file2 の結果は “My Inspector XE results – file2/r000sc” というように別々のディレクトリーに作成されます。2 つの結果のディレクトリーは同じ親ディレクトリーの下に作成されます。

inspxe-runsc には、結果の作成場所を指定するための `-result-dir` (`-r`) コマンドライン・スイッチがあります。このスイッチの動作が変更されました。以前は、r000sc のように結果が作成されるディレクトリーの名前を指定していましたが、現在は、“My Inspector XE Results - name” のように結果が作成されるディレクトリーの親ディレクトリーを指定します。つまり、–r スイッチのディレクトリー名は、結果の生成される場所から 2 つ上のディレクトリーのものになります。

inspxe-runsc のこの変更により、結果ディレクトリーが効率良く移動します。この変更に伴い、ユーザーによる対応が必要になります。–r スイッチを指定して inspxe-runsc を呼び出すスクリプトを使用している場合は、新しい動作に合わせて、-r スイッチの引数を変更してください。また、新しいバージョンの inspxe-runsc によって生成されるスタティック解析結果が、以前のバージョンの inspxe-runsc によって生成された結果と同じディレクトリーに保存されることがないように、以前の結果ファイルを新しいディレクトリーに移動する必要があります。以前のバージョンの inspxe-runsc でリンクステップが 1 つのみのビルド仕様を実行した結果は、“My Inspector XE results – name” という形式のディレクトリーに移動します。この操作を行わないと、新しく作成される結果ですべての問題が “新規” として表示されます。以前のバージョンの inspxe-runsc で複数のリンクステップを含むビルド仕様を実行した場合、スタティック解析ではさまざまな問題がありましたが、これらの問題は新しいバージョンを使用することで解決されます。この場合、以前の結果のうち最も新しいものを “My Inspector XE results – name” という形式の新しいディレクトリーに (1 つのディレクトリーに 1 つの結果が含まれるように) コピーします。これにより、新しいバージョンで作成される結果に以前の問題ステート情報が正しく適用される可能性が高くなります。

3.3 新規および変更されたコンパイラー・オプション

コンパイラー・オプションの詳細に関しては、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.3.1 インテル® Composer XE 2011 Update 6 の新規および変更されたコンパイラー・オプション

- -march=core-avx2
- -xCORE-AVX2
- -axCORE-AVX2
- -march-core-avx-i

- -xCORE-AVX-I
- -axCORE-AVX-I
- -xSSSE3_ATOM
- -fms-dialect
- -masm
- -fasynchronous-unwind-tables
- -opt-mem-layout-trans
- -fopenmp
- -parallel-source-info
- -fgnu89-inline
- -gdwarf-3
- -fno-merge-debug-strings
- -sox
- -fstack-protector-all

3.3.2 インテル® Composer XE 2011 の新規および変更されたコンパイラー・オプション

- -ansi-alias-check
- -auto-p32
- -cilk-serialize
- -diag-sc-dir
- -ffriend-injection
- -fzero-initialized-in-bss
- -fimf-absolute-error
- -fimf-accuracy-bits
- -fimf-arch-consistency
- -fimf-max-error
- -fimf-precision
- -fp-trap
- -fp-trap-all= mode[,mode,...]
- -fvar-tracking
- -fvar-tracking-assignments
- -guide
- -guide-data-trans
- -guide-file
- -guide-file-append
- -guide-opts=*string*
- -guide-par
- -guide-vec
- -intel-extensions
- -opt-args-in-regs
- -opt-matmul
- -prof-value-profiling
- -profile-functions
- -profile-loops
- -regcall
- -simd
- -Wremarks
- -Wsign-compare
- -Wstrict-aliasing

廃止予定のコンパイラー・オプションのリストは、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.4 その他の変更

3.4.1 コンパイラー環境の設定

コンパイラー環境は、compilervars.sh スクリプトを使用して設定します。
`compilervars.csh` も提供されます。

コマンドの形式は以下のとおりです。

```
source <install-dir>/bin/compilervars.sh argument
```

`argument` にはターゲット・アーキテクチャーに応じて、`ia32` または `intel64` を指定します。コンパイラー環境を設定すると、インテル® デバッガー、インテル® パフォーマンス・ライブラリー、インテル® Fortran コンパイラー (インストールされている場合) の環境も設定されます。

3.4.2 デフォルトの命令セットがインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) を必要とするものに変更

IA-32 アーキテクチャー向けのコンパイルでは、`-msse2` (旧: `-xW`) がデフォルトです。`-msse2` でビルドされたプログラムは、インテル® Pentium® 4 プロセッサーや特定のインテル以外のプロセッサーなど、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) をサポートするプロセッサー上で実行する必要があります。互換性を保証するランタイム・チェックは行われません。プログラムがサポートされていないプロセッサーで実行されている場合は、無効な命令フォルトが発生する場合があります。これにより、インテル® SSE 命令が x87 命令の代わりに使用され、高い精度ではなく、宣言された精度で計算が行われることがあるため、浮動小数点結果が変更される可能性があることに注意してください。

すべてのインテル® 64 アーキテクチャー・プロセッサーでインテル® SSE2 がサポートされています。

汎用 IA-32 の以前のデフォルトを使用する場合は、-mia32 を指定してください。

3.4.3 OpenMP* レガシー・ライブラリーの削除

本リリースでは、OpenMP* のレガシー・ライブラリーが削除されました。"互換性がある" ライブラリーのみ提供されます。

3.5 既知の問題

3.5.1 __GXX_EXPERIMENTAL_CXX0X__ マクロの未サポート

GNU* 4.6 以降の環境で `-std=c++0x` または `-std=gnu++0x` オプションを使用すると、次のような診断が出力されることがあります。

```
This file requires compiler and library support for the upcoming ISO C++
standard, C++0x. This support is currently experimental, and must be
enabled with the -std=c++0x or -std=gnu++0x compiler options. (このファイルには、新しい ISO C++ 規格である C++0x 用のコンパイラーとライブラリーのサポートが必要です。現在このサポートはまだ試験段階であり、-std=c++0x または -std=gnu++0x コンパイラー・オプションと一緒に指定する必要があります。)
```

`__GXX_EXPERIMENTAL_CXX0X__` マクロは、C++ 標準ライブラリー・ヘッダーのマクロで有効にされる一部の C++0x 機能をまだサポートしていないため、インテル® コンパイラーでは、現在 gcc 4.6 モードでこのマクロを定義していません。このため、`-std=c++0x` または `-std=gnu++0x` モードで C++ 標準ライブラリーを使用すると、g++ との互換性問題が発生することがあります。

インテル® C++ Composer XE 2011 の以前の Update リリースでは、gcc 4.3、4.4、4.5 でもこの問題がありましたが、Update 6 では解決しています。gcc 4.6 よりも前のバージョンで提供されるヘッダーファイルでは、この問題は発生しません。

3.5.2 インテル® Cilk™ Plus の既知の問題

リンクエラー “undefined reference to `__cilkrts_*' (`__cilkrts_*' への未定義の参照です。)”

2.17 よりも古いバージョンの binutils を使用している場合、インテル® Cilk™ Plus コードで上記のようなリンクエラーが発生することがあります。

```
undefined reference to `__cilkrts_get_tls_worker'
(`__cilkrts_get_tls_worker' への未定義の参照です。)
```

これは、2.17 よりも古いバージョンの binutils では、インテル® Cilk™ Plus のランタイム・ライブラリーが自動でリンクできないためです。この問題は、binutils を新しいバージョンにアップデートするか、リンカーのコマンドラインで -lcilkrts を使用してランタイム・ライブラリーを手動でリンクすることで回避できます。

3.5.3 ガイド付き自動並列化の既知の問題

プログラム全体のプロシージャー間の最適化 (-ipo) が有効な場合、単一ファイル、関数名、ソースコードの指定範囲に対してガイド付き自動並列化 (GAP) 解析は行われません。

3.5.4 スタティック・セキュリティー解析の既知の問題

3.5.4.1 仮想関数を含む C++ クラスに対する正しくないメッセージ

スタティック・セキュリティー解析機能を使用するためには、インテル® Inspector XE も必要です。

プログラムで仮想関数を含む C++ クラスが使用されている場合に、スタティック・セキュリティー解析は正しくない診断を多数出力します。場合によっては、診断結果の数が多すぎて結果ファイルが使用できないこともあります。

このような C++ ソース構造を使用しているアプリケーションでは、次のコマンドライン・スイッチを追加することで不要なメッセージを表示しないようにできます: `/Qdiag-disable:12020,12040` (Windows) または `-diag-disable 12020,12040` (Linux)。**このスイッチは、スタティック・セキュリティー解析の結果が作成されるリンクステップで追加する必要があります。**コンパイルステップで追加しただけでは十分な効果が得られません。

ビルド仕様ファイルを使用してスタティック・セキュリティー解析を行う場合は、`-disable-id 12020,12040` スイッチを inspxe-runsc の呼び出しに追加します。

例:

```
inspxe-runsc -spec-file mybuildspec.spec -disable-id 12020,12040
```

この問題を含む作成済みのスタティック・セキュリティー解析結果がある場合は、インテル® Inspector XE の GUI でそのファイルを開いて、次の手順に従って不要なメッセージを非表示にすることができます。

- 不要なメッセージは “`Arg count mismatch` (引数の数の不一致)” と “`Arg type mismatch` (引数の型の不一致)” です。それぞれの問題に対して、次の手順を実行します。
- 問題フィルターで不要な問題の種類をクリックします。これにより、それ以外の問題が非表示になります。
- 問題セットの表で任意の問題をクリックします。

- Ctrl+A キーを押すとすべての問題を選択できます。
- 右クリックしてポップアップ・メニューから [Change State (ステートの変更)] > [Not a problem (問題なし)] を選択し、不要なすべての問題のステートを設定します。
- 問題の種類のフィルターを [All (すべて)] に戻します。
- 他の不要な問題の種類に対して、上記の手順を行います。
- [Investigated/Not investigated (調査済み/未調査)] フィルターを [Not investigated (未調査)] に設定します。このフィルターは最後の方にあるため、フィルターペインを下にスクロールしないと見えないことがあります。[Not a problem (問題なし)] ステートは [Not investigated (未調査)] と見なされるため、これで不要なメッセージが非表示になります。

4 インテル® デバッガー (IDB)

次の注意事項は、IA-32 アーキテクチャー・システムおよびインテル® 64 アーキテクチャー・システムで実行するインテル® デバッガー (IDB) のグラフィカル・ユーザー・インターフェイス (GUI) についてです。このバージョンでは、idb コマンドは GUI を起動します。コマンドライン・インターフェイスを起動するには、idbc を使用します。

4.1 Java* ランタイム環境の設定

インテル® IDB デバッガーのグラフィカル環境は、Java* アプリケーションで構築されており、実行には Java* ランタイム環境 (JRE) が必要です。デバッガーは、5.0 (1.5) または 6.0 (1.6) JRE をサポートしています。

配布元の手順に従って JRE をインストールします。

最後に、JRE のパスを設定する必要があります。

```
export PATH=<path_to_JRE_bin_dir>:$PATH
```

4.2 デバッガーの起動

デバッガーを起動するには、まず始めに、「コンパイラー環境の設定」で説明されているコンパイラー環境が設定されていることを確認してください。その後、次のコマンドを使用します。

```
idb
```

または

```
idbc
```

(必要に応じて)

GUI が開始され、コンソールウィンドウが表示されたら、デバッグセッションを開始できます。

注: デバッグする実行ファイルが、デバッグ情報付きでビルドされ、実行可能ファイルであることを確認してください。必要に応じて、アクセス権を変更します。

例: `chmod +x <application_bin_file>`

4.3 その他のドキュメント

インテル® コンパイラー / インテル® デバッガー・オンライン・ヘルプ は、デバッガーのグラフィカル・ユーザー・インターフェイスの [Help (ヘルプ)] > [Help Contents (ヘルプ目次)] で表示できます。

[Help (ヘルプ)] ボタンが表示されているデバッガーのダイアログから状況依存ヘルプにもアクセスできます。

4.4 デバッガー機能

4.4.1 IDB の主な機能

デバッガーは、インテル® IDB デバッガーのコマンドライン・バージョンのすべての機能をサポートしています。デバッガー機能は、デバッガー GUI または GUI コマンドラインから呼び出すことができます。グラフィカル環境を使用する場合は、既知の制限を参照してください。

4.4.2 インテル® Inspector XE 2011 Update 6 による IDB の “break into debug” のサポート

インテル® Inspector XE 2011 Update 6 は、インテル® Composer XE 2011 Update 6 に含まれるインテル® デバッガーによる “break into debug” モードをサポートしています。詳細は、インテル® Inspector XE 2011 のリリースノートを参照してください。

4.5 既知の問題と変更点

4.5.1 Pardus* システムのデフォルトの .gdbinit スクリプトでデバッガーがクラッシュ

Pardus* システムで `idbc` または `idb` を開始したときにデバッガーがクラッシュする場合は、オプション `-nx` を追加してデフォルトの `.gdbinit` スクリプトを使用しないようにしてください。

4.5.2 Pardus* システムでスレッド情報が利用できない

Pardus* システムのデフォルトの `libthread_db.so` ライブラリーの問題により、デバッガーでマルチスレッド・アプリケーションをデバッグしたときにスレッド情報が検出できません。

4.5.3 Thread Data Sharing Filters (スレッドデータ共有フィルター) が正しく動作しない

Thread Data Sharing Filters (スレッドデータ共有フィルター) を設定すると、デバッガーが予期しない動作をすることがあります。スレッドはデータ共有検出の後に続行せず、デバッガーは SIG SEGV で終了します。

フィルターが有効な状態でデータ共有検出に関連する問題が発生した場合は、[Thread Data Sharing Filters (スレッドデータ共有フィルター)] ウィンドウのコンテキスト・メニューでフィルターをすべて無効にしてください。

4.5.4 コアファイルのデバッグ

コアファイルをデバッグするには、以下のようにコマンドライン・オプションを指定してデバッガー (コマンドライン・デバッガー idbc または GUI デバッガー idb) を開始する必要があります。

```
idb|idbc <executable> <corefile>
```

または

```
idb|idbc <executable> -core <corefile>
```

コアファイルのデバッグを開始すると、デバッガーはライブプロセス (例えば、新しいプロセスのアタッチや作成) をデバッグできません。また、ライブプロセスをデバッグしているときはコアファイルをデバッグできません。

4.5.5 シェルで $HOME が設定されていないとデバッガーがクラッシュ

デバッガーを起動したシェルで $HOME 環境変数が設定されていない場合、“セグメンテーション違反” でデバッガーが終了します。

4.5.6 コマンドライン・パラメーター -idb と -dbx は未サポート

デバッガーのコマンドライン・パラメーター -idb と -dbx は、デバッガー GUI ではサポートされていません。

4.5.7 プロセッサーのデバッグレジスター (ハードウェア・ベース) を使用したウォッチポイント (インテル® Composer XE 2011 Update 6)

インテル® Composer XE 2011 Update 6 (IDB 12.1) では、プロセッサーのデバッグレジスターを使用したウォッチポイントが完全にサポートされています。設定方法は、使用するプロセッサー・アーキテクチャーにより異なります。IA-32 およびインテル® 64 アーキテクチャー・システムでは次の制限があります (可能な場合、インテル® デバッガーは、適切なエラーメッセージを出力します)。

- ウォッチするメモリー領域のサイズは、1、2、4 または 8 (インテル® 64 のみ) バイトでなければなりません。
- ウォッチするメモリー領域の開始アドレスは、ウォッチするサイズでアラインされていなければなりません。例えば、ウォッチするサイズが 2 バイトの場合、開始アドレスは奇数であってはなりません。
- アクティブ/有効なウォッチポイントは最大 4 つまでサポートされています。使用されていないウォッチポイントを無効にすることで、リソースを解放したり、別のウォッチポイントを作成したり有効にすることができます。
- 次のアクセス方法のみサポートされています。
    - 書き込み: 書き込みアクセスでトリガーされます。
    - 指定: 書き込みまたは読み取りアクセスでトリガーされます。
    - 変更: 実際に値を変更した書き込みアクセスでトリガーされます。
- ウォッチするメモリー領域が複数ある場合、それぞれの領域はオーバーラップしていてはなりません。
- ウォッチポイントは、スコープには関係ありませんが、プロセスに関連付けられています。プロセスが実行中である限り、ウォッチポイントはアクティブ/有効です。プロセスが終了されると (例えば、プロセスがリターンした場合など)、ウォッチポイントは無効になります。必要に応じて、ユーザーはウォッチポイントを再度有効にすることができます。
- デバッガーを使用してウォッチするメモリー領域にアクセスすると (例えば、変数に異なる値を割り当てるなど)、ハードウェアの検出がスキップされます。そのため、ウォッチポイントは、デバッグ対象がウォッチするメモリー領域にアクセスした場合のみトリガーされます。
- デバッグ対象が仮想マシン内のゲスト OS で実行されている場合、命令やコード行をステップオーバーすると、プロセスは停止しないで継続することがあります。ウォッチポイントは、実際のハードウェアでデバッグ対象を実行した場合にのみ、動作が保証されています。

4.5.8 位置独立実行ファイル (PIE) のデバッグは未サポート

一部のシステムでは、コンパイラーは位置独立実行ファイル (PIE) を生成します。その場合、コンパイル時とリンク時の両方に -fno-pie フラグを指定する必要があります。そうでないと、アプリケーションをデバッグできません。

4.5.9 コマンドライン・パラメーター -parallel は未サポート

デバッガーのコマンドライン・パラメーター -parallel は、シェルのコマンドプロンプトおよびデバッガー GUI のコンソールウィンドウではサポートされていません。

4.5.10 [Signals (シグナル)] ダイアログが動作しない

GUI ダイアログの [Debug (デバッグ)] > [Signal Handling (シグナル処理)]、またはショートカット・キーの Ctrl+S でアクセス可能な [Signals (シグナル)] ダイアログが正しく動作しないことがあります。シグナル・コマンドライン・コマンドを代わりに使用する場合は、インテル® デバッガー (IDB) マニュアルを参照してください。

4.5.11 GUI のサイズ調整

デバッガーの GUI ウィンドウのサイズが小さくなり、一部のウィンドウが表示されていないことがあります。ウィンドウを拡大すると、隠れているウィンドウが表示されます。

4.5.12 $cdir ディレクトリー、$cwd ディレクトリー

$cdir はコンパイル・ディレクトリーです (記録されている場合)。$cdir は、ディレクトリーが設定されている場合にサポートされます。シンボルとしてサポートされるわけではありません。

$cwd は現在の作業ディレクトリーです。セマンティクスもシンボルもサポートされていません。

$cwd と '.' の違いは、$cwd はデバッグセッション中に変更された現在の作業ディレクトリーを追跡する点です。'.' は、ソースパスへのエントリーが追加されると直ちに現在のディレクトリーに展開されます。

4.5.13 info stack の使用

デバッガーコマンド `info stack` は、以下のように、負のフレームカウントの使用方法が現在 gdb とは異なります。

```
info stack [num]
```

num が正の場合は最内の num フレーム、ゼロの場合はすべてのフレーム、負の場合は最内の -num フレームを逆順で出力します。

4.5.14 $stepg0 のデフォルト値の変更

デバッガー変数 `$stepg0` のデフォルト値が `0` に変更されました。値 "0" の設定では、"step" コマンドを使用する場合、デバッガーはデバッグ情報なしでコードにステップオーバーします。以前のデバッガーバージョンと互換性を保つようにするには、次のようにデバッガー変数を 1 に設定します。

```
(idb) set $stepg0 = 1
```

4.5.15 一部の Linux* システムでの SIGTRAP エラー

一部の Linux* ディストリビューション (例: Red Hat* Enterprise Linux* Server 5.1 (Tikanga)) では、デバッガーがブレークポイントで停止した後、ユーザーがデバッグを続行すると SIGTRAP エラーが発生することがあります。この問題を回避するには、SIGTRAP シグナルを次のようにコマンドラインで定義します。

```
(idb) handle SIGTRAP nopass noprint nostop
SIGTRAP is used by the debugger.
SIGTRAP       No        No      No              Trace/breakpoint trap
(idb)
```

警告: この回避策は、デバッグ対象にシグナルを送信するすべての SIGTRAP をブロックします。

4.5.16 MPI プロセスのデバッグには idb GUI は使用不可

MPI プロセスのデバッグに idb GUI を使用することはできません。コマンドライン・インターフェイス (idbc) を使用してください。

4.5.17 GUI でのスレッド同期ポイントの作成

単純なコードやデータのブレークポイントでは [Location (場所)] が必須です。スレッド同期ポイントでは [Location (場所)] と [Thread Filter (スレッドフィルター)] の両方が必須です。スレッド同期ポイントは、スレッドの同期を指定します。その他の種類のブレークポイントでは、このフィールドは作成されたブレークポイントの中からリストされているスレッドに関するものだけに制限します。

4.5.18 [Data Breakpoint (データ・ブレークポイント)] ダイアログ

[Within Function (関数内)] フィールドと [Length (長さ)] フィールドは使用されていません。ウォッチする場所は、ウォッチする長さを暗黙的に提供します (効率的な式の型が使用されます)。また、[Read （読み取り)] アクセスも利用できません。

4.5.19 IA-32 アーキテクチャー向けのスタック・アライメント

IA-32 アーキテクチャー向けのデフォルトのスタック・アライメントの変更に伴い、下位呼び出し (デバッグ対象のコードを実行する式の評価など) を使用すると失敗することがあります。場合によっては、デバッグ対象がクラッシュし、デバッグセッションが再起動されることもあります。この機能を使用する場合は、`-falign-stack=<mode>` オプションを使用して 4 バイトのスタック・アライメントでコードをコンパイルしてください。

4.5.20 GNOME 環境の問題

GNOME 2.28 では、デバッガーのメニューアイコンがデフォルトで表示されないことがあります。メニューアイコンを表示するには、[System (システム)] > [Preferences (設定)] > [Appearance (外観の設定)] > [Interface (インターフェイス)] タブで [Show icons in menus (メニューにアイコンを表示)] を有効にします。[Interface (インターフェイス)] タブがない場合は、次のようにコンソールで GConf キーを使用してこの変更を行うことができます。

```
gconftool-2 --type boolean --set
/desktop/gnome/interface/buttons_have_icons true

gconftool-2 --type boolean --set /desktop/gnome/interface/menus_have_icons
true
```

4.5.21 オンラインヘルプへのアクセス

システムで IDB デバッガー GUI の [Help (ヘルプ)] メニューからオンラインヘルプにアクセスできない場合は、次の Web ベースのドキュメントを利用できます。

http://intel.ly/o5DMp9

5 Eclipse* 統合

インテル® C++ コンパイラーでは、Eclipse* 機能と関連プラグイン (インテル® C++ Eclipse* 製品拡張) がインストールされます。これらを Eclipse* 統合開発環境 (IDE) として追加すると、インテル® C++ コンパイラーが Eclipse* でサポートされます。これにより、インテル® C++ コンパイラーを Eclipse* 統合開発環境から使用して、アプリケーションを開発することができます。

5.1 提供されている統合

Eclipse* プラットフォームのバージョン 3.6 用のファイルは次のディレクトリーにあります。

```
<install-dir>/eclipse_support/cdt7.0/eclipse
```

統合には、Eclipse* プラットフォームのバージョン 3.6、Eclipse* C/C++ Development Tools (CDT) のバージョン 7.0 以降、および Java* ランタイム環境 (JRE) 5.0 (1.5) または 6.0 (1.6) が必要です。

Eclipse* プラットフォームのバージョン 3.7 用のファイルは次のディレクトリーにあります。

```
<install-dir>/eclipse_support/cdt8.0/eclipse
```

統合には、Eclipse* プラットフォームのバージョン 3.7、Eclipse* C/C++ Development Tools (CDT) のバージョン 8.0 以降、および Java* ランタイム環境 (JRE) 6.0 (1.6) Update 11 が必要です。

5.1.1 統合に関する注意事項

すでに適切なバージョンの Eclipse*、CDT、および JRE が環境にインストールされ、設定されている場合は、このセクションの「Eclipse でのインテル® C++ Eclipse 製品拡張のインストール方法」で説明するように、インテル® C++ Eclipse* 製品拡張を Eclipse* に追加インストールできます。そうでない場合は、このセクションの「Eclipse*、CDT、および JRE の入手方法とインストール方法」で説明するように、最初に Eclipse*、CDT、および JRE を入手して、インストールしてください。そして、その後にインテル® C++ Eclipse* 製品拡張をインストールします。

5.2 Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法

既存の Eclipse* の構成にインテル® C++ Eclipse* 製品拡張を追加するには、Eclipse* から次の手順を実行します。

[Help (ヘルプ)] > [Install New Software... (新規ソフトウェアのインストール...)] を選択して [Available Software (利用可能なソフトウェア)] ページを開きます。[Add... (追加...)] ボタンをクリックし、[Local... (ローカル...)] を選択します。ディレクトリー・ブラウザーが開きます。インテル® C++ コンパイラーのインストール・ディレクトリーにある eclipse ディレクトリーを選択します。例えば、root としてコンパイラーをデフォルトのディレクトリーにインストールした場合は、`/opt/intel/composer_xe_2011_sp1.<n>.<xxx>/eclipse_support/cdt7.0/eclipse` を選択します。(ここでは CDT 7.0 を使用していると仮定しています。) CDT 8.0 を使用している場合は、パスの "`cdt7.0`" を "`cdt8.0`" に置換します。[OK] をクリックして、ディレクトリー・ブラウザーを閉じます。[OK] をクリックして、[Add Site (サイトの追加)] ダイアログを閉じ、インテル® C++ 統合機能の 2 つのボックスを選択します。1 つめは [Intel® C++ Compiler Documentation (インテル® C++ コンパイラー・ドキュメント)]、2 つめは [Intel® C++ Compiler XE 12.1 for Linux* OS (インテル® C++ コンパイラー XE 12.0 Linux* 版)] です。注:[Group items by category (項目をカテゴリー別にグループ化)] がオンの場合、インテルの機能は表示されません。インテルの機能を表示するには、このオプションをオフにします。インテル® デバッガー (idb) と Eclipse* 製品拡張もインストールした場合は、同じ方法で idb 製品拡張サイトを Eclipse* に追加することで、Eclipse* 内で idb を使用できるようになります。

[Next (次へ)] ボタンをクリックします。[Install (インストール)] ダイアログが表示され、インストールする項目を確認できます。[Next (次へ)] をクリックします。契約に同意するかどうかを確認するメッセージが表示されます。契約に同意したら、[Finish (完了)] をクリックします。署名されていないコンテンツを含むソフトウェアをインストールしようとしていることを示す [Security Warning (セキュリティの警告)] ダイアログが表示されたら [OK] をクリックします。これで、インストールが開始します。

Eclipse* の再起動を求められたら [Yes (はい)] を選択します。Eclipse* が再起動したら、インテル® C++ コンパイラーを使用する CDT プロジェクトを作成して作業することができます。詳細は、インテル® C++ コンパイラーのドキュメントを参照してください。インテル® C++ コンパイラーのドキュメントは、[Help (ヘルプ)] > [Help Contents (ヘルプ目次)] > [Intel(R) C++ Compiler XE 12.1 User and Reference Guides (インテル® C++ コンパイラー XE 12.1 ユーザー・リファレンス・ガイド)] で表示できます。

5.2.1 Eclipse* へのインテル® デバッガーの統合

上記の手順を実行して Eclipse* を再起動してから、次の手順に従って Eclipse* にインテル® デバッガーを統合します。

1. [Run (実行)] > [Debug Configurations... (デバッグ設定...)] を選択してデバッグの起動設定を作成します。
2. 表示されるダイアログボックスで [C/C++ Application (C/C++ アプリケーション)] を右クリックして [New (新規)] を選択します。
3. 右側にいくつかのタブが表示されます。右下に [Using GDB (DSF) Create Process Launcher - Select other... (GDB (DSF) プロセス作成ランチャーの使用 - その他の選択...)] というラベルが表示されます。これをクリックするとダイアログが表示されます。[Standard Create Process Launcher (標準プロセス作成ランチャー)] を選択して [OK] をクリックします。
4. [Debugger (デバッガー)] タブでコンボボックスからインテル® デバッガー (idbc) を選択します。idbc を idbc. へのフルパスに置換します。

5.3 Eclipse*、CDT、および JRE の入手方法とインストール方法

Eclipse* は Java* アプリケーションのため、実行には Java* ランタイム環境 (JRE) が必要です。JRE は、オペレーティング環境 (マシン・アーキテクチャー、オペレーティング・システムなど) に応じてバージョンを選択します。また、多くの JRE の中から選択可能です。

Eclipse* 3.7 および CDT 8.0 の両方が含まれたパッケージは、以下の Web サイトから入手できます。

http://www.eclipse.org/downloads/

スクロールして、“Eclipse IDE for C/C++ Developers” を確認してください。必要に応じて、Linux* 32 ビットまたは Linux* 64 ビットをダウンロードしてください。

以前の Eclipse 3.6 + CDT 7.0 プラットフォームをサポートする必要がある場合は、「Related Links」の「Eclipse Helios (3.6)」 > 「Download Helios」を参照してください。同じページに「Eclipse IDE for C/C++ Developers」の情報もあります。必要に応じて、Linux* 32 ビットまたは Linux* 64 ビットをダウンロードしてください。

5.3.1 JRE、Eclipse*、CDT のインストール

適切なバージョンの Eclipse*、CDT、および JRE をダウンロードしたら、次の手順に従ってインストールします。

1. 配布元の手順に従って、JRE をインストールします。
2. Eclipse* をインストールするディレクトリーを作成し、`cd` でこのディレクトリーに移動します。ここでは、このディレクトリーを `<eclipse-install-dir>` と表記します。
3. Eclipse* パッケージのバイナリー、`.tgz` ファイルを `<eclipse-install-dir>` ディレクトリーにコピーします。
4. `.tgz` ファイルを展開します。
5. `eclipse` を起動します。

これで、Eclipse* の構成にインテル® C++ 製品拡張を追加する準備が完了です。追加する方法は、「Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法」のセクションで説明されています。Eclipse の初回起動時の設定については、次のセクションを参照してください。

5.4 インテル® C++ コンパイラーで開発するための Eclipse* の起動

LANG 環境変数を設定していない場合は、設定してください。次に例を示します。

```
setenv LANG ja_JP.UTF8
```

Eclipse* を起動する前に compilervars.csh (または .sh) スクリプトを実行して、インテル® C++ コンパイラー関連の環境変数を設定します。

```
source <install-dir>/bin/iccvars.csh arch_arg
("arch_arg" は "ia32" または "intel64" のいずれか)
```

Eclipse* を実行するには JRE が必要なため、Eclipse* を起動する前に JRE が利用可能であることを確認してください。PATH 環境変数の値をシステムにインストールされている JRE の java ファイルのフォルダーへのフルパスに設定するか、Eclipse* コマンドの -vm パラメーターでシステムにインストールされている JRE の java 実行ファイルへのフルパスを参照します。

例:

```
eclipse -vm /JRE folder/bin/java
```

Eclipse* がインストールされているディレクトリーから Eclipse* 実行ファイルを直接起動します。次に例を示します。

```
<eclipse-install-dir>/eclipse/eclipse
```

5.5 Fedora* システムでのインストール

root アカウントではなくローカルアカウントとして、インテル® C++ コンパイラー Linux* 版を Fedora* 搭載の IA-32 またはインテル® 64 システムにインストールすると、Eclipse* を起動する際に、コンパイラーまたはデバッガーで Eclipse* グラフィカル・ユーザー・インターフェイスが正しく表示されないことがあります。この場合、通常、JVM Terminated エラーが表示されます。また、システムレベルの root アカウントでソフトウェアをインストールし、それ以下の権限のユーザーアカウントで実行する場合もエラーが発生します。

これは、Fedora* に実装されているセキュリティーのレベルが低いためです。この新しいセキュリティーは、ダイナミック・ライブラリーなど、システムリソースへのアクセスに悪影響を及ぼすことがあります。一般ユーザーがコンパイラーを使用するためには、システム管理者は SELinux セキュリティーを調整する必要があります。

5.6 コンパイラー・バージョンの選択

Eclipse* プロジェクトでは、異なるバージョンのインテル® C++ コンパイラーがインストールされている場合、コンパイラーのバージョンを選択できます。IA-32 アーキテクチャー・システムでサポートされているインテル® コンパイラーのバージョンは、9.1、10.0、10.1、11.0、11.1、12.1 です。インテル® 64 アーキテクチャー・システムでは、コンパイラー・バージョン 11.0、11.1、12.1 がサポートされています。

6 インテル® インテグレーテッド・パフォーマンス・プリミティブ

このセクションでは、インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) のこのバージョンでの変更点、新機能、および最新情報をまとめています。インテル® IPP についての詳細は、次のリンクを参照してください。

- **新機能:** インテル® IPP 製品ページ (http://intel.ly/o6nf00 (英語)) およびインテル® IPP リリースノート (http://intel.ly/pSaf2j (英語)) を参照してください。
- **ドキュメント、ヘルプ、サンプル:** インテル® IPP 製品ページ (http://intel.ly/o6nf00) のドキュメントのリンクを参照してください。

6.1 別途ダウンロード可能なインテル® IPP 暗号化ライブラリー

インテル® IPP 暗号化ライブラリーは別途ダウンロード可能です。ダウンロードとインストールの手順については、http://intel.ly/ndrGnR (英語) を参照してください。

6.2 インテル® IPP コードサンプル

インテル® IPP コードサンプルは、以下の Web サイトから入手できます。
http://intel.ly/pnsHxc (英語)

サンプルには、オーディオ/ビデオコーデック、画像処理、メディア・プレーヤー・アプリケーション、C++/C#/Java* からの呼び出し関数のソースコードが含まれています。サンプルのビルド方法についての説明は、各サンプルのインストール・パッケージの `readme` ファイルをご覧ください。

7 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® マス・カーネル・ライブラリー (インテル® MKL) の変更点、新機能、および最新情報をまとめています。問題の修正については、 http://intel.ly/rITU0r (英語) を参照してください。

7.1 注意事項

- インテル® MKL に含まれる GMP* 数学関数は、将来のリリースでは削除されます。
- タイミング関数 mkl_set_cpu_frequency() は古い関数です。代わりに、『インテル® MKL リファレンス・マニュアル』で説明されている mkl_get_max_cpu_frequency()、mkl_get_clocks_frequency()、mkl_get_cpu_frequency() を使用してください。
- PARDISO ドメインを指定するために定義されている MKL_PARDISO 定数は、mkl_domain_set_num_threads() 関数で使用できなくなりました。代わりに、MKL_DOMAIN_PARDISO を使用してください。
- 畳み込みルーチンと相関ルーチンは、将来のリリースでは 10.2 Update 3 との下位互換性はなくなります。

7.2 本バージョンでの変更

7.2.1 最初のリリースでの変更

- BLAS
  - 一度に 2 つの行列-ベクトル積を計算するための新しい関数: [D/S]GEM2VU、[Z/C]GEM2VC
  - 混合精度の一般的な行列-ベクトル積を計算するための新しい関数: [DZ/SC]GEMV
  - 2 つのスケールされたベクトルの和を計算するための新しい関数: *AXPBY
  - 主要関数においてインテル® AVX による最適化: SMP LINPACK、レベル 3 BLAS、DDOT、DAXPY

- LAPACK
  - 行優先順に対応した LAPACK 用の C インターフェイス
  - 1 つの新しい計算ルーチン (*GEQRFP)、2 つの新しい補助ルーチン (*GEQR2P と *LARFGP)、LAPACK 3.2.1 のアップデートを含む Netlib LAPACK 3.2.2 との統合
  - 主要関数においてインテル® AVX による最適化: DGETRF、DPOTRF、DGEQRF
- PARDISO
  - マルチコア環境で問題と解のステップのパフォーマンスが向上
  - スパースの右辺の解算出と部分解ベクトルを出力する部分解算出の追加
  - アウトオブコア (OOC) 因数分解のパフォーマンスが向上
  - ゼロベース (C スタイル) の配列インデックスのサポート
  - 対称行列のスパースデータ構造で行列の対角上のゼロが不要
  - 新しい ILP64 PARDISO インターフェイスにより、LP64 ライブラリーにリンクされている場合に LP64 と ILP64 の両バージョンを使用可能
  - OOC モードでディスクにファイルを格納するのに必要なメモリーを並べ替え直後に予測可能
- スパース BLAS
  - 形式変換関数ですべてのデータ型に対応 (単精度/倍精度の実数/複素数データ)、および関数の戻り値として並べ替えあり/並べ替えなし配列を使用可能
- FFT
  - 新しい MPI FFTW 3.3alpha1 ラッパーによる新しいクラスター機能
  - クラスター FFT のロードバランスの改善によりパフォーマンスが向上
  - すべての 1D/2D/3D FFT においてインテル® AVX による最適化
  - SSE4.2 命令セットをサポートするすべてのシステムにおいて、基数が混在する単精度/倍精度データの 2D/3D FFT のパフォーマンスが向上
  - 2D/3D FFT における 2 つの実数配列として表される分割複素数データのサポート
  - 長さが大きな素数である 1D 複素数-複素数変換のサポート
  - クラスター 1D 複素数変換のハイブリッド並列化 (MPI + OpenMP)、および (MPI プロセス数の倍数である) ベクトル長のパフォーマンスの向上
- VML
  - (ax+b)/(cy+d) の計算を行うための新しい関数。a、b、c、d はスカラー、x、y は実数ベクトル: v[s/d]LinearFrac()
  - 主要関数においてインテル® AVX による最適化
  - デノーマル数をゼロに設定するための新しいモデル、複素ベクトルのオーバーフロー・サポート、各 VML 関数に対して精度を設定するための追加パラメーターを含む新しい関数
- VSL
  - 新しいサマリー統計関数群。基礎統計、共分散/相関関係、プールされたグループ/部分/厳密な共分散/相関関係、分位数/変量分位数、外れ値検出アルゴリズム、欠測値をサポート
    - パフォーマンスが最適化されたアルゴリズム:欠測値をサポートするための MI アルゴリズム、厳密な共分散を計算するための TBS アルゴリズム、外れ値を検出するための BACON アルゴリズム、(変量データの) 分位数を計算するための ZW アルゴリズム、プールされた共分散を計算するための 1PASS アルゴリズム
  - SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
  - インテル® AVX による最適化: MT19937 と MT2203 BRNG
- ランタイムにディスパッチされるダイナミック・ライブラリーの追加により、ランタイムに検出された CPU またはライブラリー関数呼び出しに応じて、依存性のあるライブラリーを動的にロードする単一のインターフェイス・ライブラリーへのリンクが可能
- カスタム・ダイナミック・ライブラリー・ビルダーは、Linux* および Mac OS* X オペレーティング・システムにおいてランタイムにディスパッチされるライブラリーを使用

- 新しいディレクトリー構造により、インテル® MKL ライブラリーとインテル® Parallel Studio XE 製品ファミリーの統合が単純化され、これまでの "em64t" ディレクトリーが "intel64" ディレクトリーに変更
- 本リリースではインテル® Itanium® アーキテクチャー (IA-64) をサポートしていないため、IA-64 用の最新リリースはインテル® MKL 10.2
- スパースソルバー機能をインテル® MKL のコア・ライブラリーに完全統合。また名前に "solver" を含むライブラリーを製品から削除

7.2.2 Update 1 での変更

- PARDISO/DSS: F90 オーバーロード API の追加 (詳細は、インテル® MKL リファレンス・マニュアルを参照してください)
- PARDISO: 統計情報をより見やすく改良
- スパース BLAS: 最新のインテル® プロセッサーにおいて ?BSRMM 関数のパフォーマンスが向上
- FFT: 負のストライドのサポート
- FFT サンプル: DFTI と FFTW3 の両方のインターフェイスを使用した分割複素 FFT の C および Fortran サンプルの追加
- VML: SSE2 および SSE3 対応システムにおいて、インプレースの Add/Sub/Mul/Sqr 実関数のパフォーマンスが向上
- ポアソン・ライブラリー: ポアソン・ライブラリー関数のデフォルトの動作をシリアルから並列に変更
- 問題の修正: http://intel.ly/rlTU0r (英語)

7.2.3 Update 2 での変更

- BLAS: インテル® Xeon® プロセッサー 5600 番台において転置関数のパフォーマンスが向上
- BLAS: 転置ルーチンのサンプルの追加
- FFT: 必要な精度の関数のみをリンクすることでアプリケーションのフットプリントを小さくする方法を示した Fortran サンプルの追加
- FFT: CCE ストレージを使用するインプレース実数変換にストライドの一貫性チェックを追加
- FFT: 多次元変換のスレッド化の追加
- VSL: クアッドコア インテル® Xeon® プロセッサー 5500 番台において単精度/倍精度の多変量ガウス分布乱数ジェネレーターのパフォーマンスが向上
- VML: インテル® Xeon® プロセッサー 5500 番台において Add、Mul、Sub 関数のインプレース操作のパフォーマンスが向上
- 問題の修正: http://intel.ly/rlTU0r (英語)

7.2.4 Update 3 での変更

- BLAS: インテル® Xeon® プロセッサー 5400 番台を搭載した 32 ビットの Windows* システムにおいて DSYRK、DTRSM、DGEMM のマルチスレッド・パフォーマンスが向上
- LAPACK: 対称/エルミート行列関数および補助関数における連立線形方程式ソルバーの向上、CS (余弦/正弦) 分解を含む Netlib LAPACK 3.3 の実装
- PARDISO: 0 ベースの順列ベクトルの入力をサポート
- PARDISO: pardisoinit() ルーチンのドキュメント化
- PARDISO: 複数の右辺 (RHS) を含む PARDISO のシリアル・パフォーマンスが向上
- PARDISO: 小さな行列のパフォーマンスを向上させる解のステップの並列化の独立制御。詳細は、iparm(25) の説明を参照してください。
- PARDISO: 後方代入の減少による右辺全体の部分解計算。詳細は、iparm(31) の説明を参照してください。
- FFT: 最大 3 ～ 7 次元の実数 FFT 変換の実装
- FFT: 2 つの実数配列として表される分割複素数データを使用した多次元複素数変換の並列化

- クラスター FFT: FORTRAN 90 インターフェイスの拡張による実数-複素数変換への対応、および新しいサンプルの追加
- VML: 新しい Pack/Unpack 複素関数と Gamma/LGamma 実関数の追加
- VML: インテル® Xeon® プロセッサー 5600 番台およびインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーで、すべての関数におけるショートベクトル (< 100) の演算、すべての関数におけるアライメントされていない入力ベクトルの演算、sPow2o3 関数、拡張パフォーマンス (EP) バージョンの Add および Sub 複素関数のパフォーマンスが向上
- VSL: 乱数ジェネレーター (RNG) ストリームからメモリーへの保存、またはメモリーからの復元を行うための関数の追加
- VSL: 新しい UniformBits32 関数および UniformBits64 関数の追加
- VSL: MT2203 BRNG でサポートされる一意のストリーム数を 1024 から 6024 に拡張
- 問題の修正: http://intel.ly/rITU0r (英語)

7.2.5 Update 4 での変更

- BLAS: インテル® Xeon® プロセッサー 5400 番台以降において DTRMM のパフォーマンスが向上
- BLAS: すべての 64 ビット対応プロセッサー、特にインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーにおいて DTRSM のパフォーマンスが向上
- LAPACK: LAPACK 3.3.1 リリースでの問題の修正に対応
- OOC PARDISO: アウトオブコア処理に必要なメモリー量の推定が向上
- FFT: スレッディングの改善による 1D 実数 FFT スケーリングの向上
- FFT: 新しいシングル・ダイナミック・ライブラリー・リンク・モデルを使用するように C および Fortran の FFT サンプルを更新
- VML: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、すべての精度で、Hypot 実関数と Abs 複素関数の単精度のパフォーマンス拡張バージョン、および Arg、Div、Mul、MulByConj 複素関数のパフォーマンスが向上
- サービス関数: インテル® MKL のサービス関数の追加と拡張 (詳細は、http://software.intel.com/en-us/articles/intel-mkl-103-release-notes/ (英語) のオンライン・リリースノートを参照してください)
- 問題の修正: http://intel.ly/rITU0r (英語)

7.2.6 Update 5 での変更

- BLAS: パフォーマンスの向上: {S,C,Z}TRSM (インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサー); {S,D}GEM2VU (インテル® AVX 対応プロセッサー、インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー 5500 番台)
- BLAS: スケーリングの向上: ?TRMV (すべてのアーキテクチャーの大規模行列); DGEMM (インテル® Xeon® プロセッサー 5400 番台でスレッド数が奇数の場合)
- LAPACK: LAPACK 3.3.1 の拡張とそれぞれの LAPACKE インターフェイスに対応
- LAPACK: 一般固有値問題に使用される ?SYGST と ?HEGST のパフォーマンスが向上
- LAPACK: LU 分解された行列の逆行列のパフォーマンスが向上 (?GETRI)
- PARDISO: 転置と共役転置の解算出 (ATx=b と AHx=b) の追加; 圧縮されたスパース列 (CSC) 形式のサポート
- PARDISO: MKL_PARDISO_OOC_MAX_SWAP_SIZE 環境変数とインコア PARDISO を使用して、必要なメモリー量が利用可能なメモリー量をやや上回る場合に、アウトオブコア PARDISO のパフォーマンスが向上
- 最適化ソルバー: RCI Trust-Region ソルバーに Inf と NaN のチェックを追加
- FFT: インテル® SSE3 以降に対応したインテル® プロセッサーにおいて、サポートされているすべての精度で 2x2x2 から 10x10x10 の小さな立方体に対する 3D FFT のパフォーマンスが向上

- FFT サンプル: インテル® MKL の DFTI と FFTW の一般的な使用例を示すサンプルコードを変更
- VSL: インテル® Core™ i7-2600 プロセッサーを搭載した 64 ビットのオペレーティング・システムのマシンにおいて、単精度の MT19937 および MT2203 基本乱数ジェネレーターのパフォーマンスが向上
- VSL: SOBOL 準乱数ジェネレーターの整数バージョンのパフォーマンスが向上。

7.2.7 Update 6 での変更

- スパース BLAS: mkl_?csrbsr 変換関数に BSR 形式から CSR 形式への変換時にゼロの要素を検出し削除する新しいオプションを追加
- 問題の修正: http://intel.ly/rITU0r (英語)

7.2.8 Update 7 での変更

- BLAS: 最近のすべてのインテル® Xeon® プロセッサーにおいて、出力行列が小さく外積が大きい (つまり、入力行列が矩形) の場合に DSYRK/SSYRK のマルチスレッド・パフォーマンスが向上
- BLAS: 最近のすべてのインテル® Xeon プロセッサーにおいて、小さな問題 (<10、beta =1) で ?GEMM のパフォーマンスが向上
- BLAS: 32 ビットのプログラムを実行するインテル® Xeon プロセッサー 5500/5600/7500 番台 において、INCX=1 の小さな問題で DSCAL のパフォーマンスが向上
- BLAS のような拡張: 正方行列のインプレース転置のキャッシュ効率とスレッディングの向上
- PARDISO: PARDISO 用の独立したスレッド化コントロールの追加; mkl_domain_set_num_threads() 関数での MKL_DOMAIN_PARDISO の使用
- ポアソン・ライブラリー: 2D/3D 周期的境界条件のサポートの追加
- ドキュメント・ディレクトリーへのリンク・ライン・アドバイザーの追加
- libtool などのスクリプトツールとともに使用するコマンドライン・リンク・ツールの追加
- 次のコンポーネントの関数の stdcall プロトタイプを含む C ヘッダーファイルの追加:BLAS、スパース BLAS、LAPACK、PARDISO/DSS、RCI 反復ソルバー、ベクトル数学関数、ベクトル・スタティスティカル関数、およびサポート関数
- mkl_domain_set_num_threads() 関数でドメインの指定に使用する定数の名前を変更 (例: MKL_BLAS は MKL_DOMAIN_BLAS に変更); MKL_PARDISO を除き、古い名前も継続して使用可能
- 問題の修正: http://intel.ly/rITU0r (英語)

7.2.9 Update 8 での変更

- データ適合コンポーネント: ベクトルスプラインの構築、セル探索や二分探索、スプライン補間の評価、微分、積分の 1 次元アルゴリズムをカバーする新しいデータ適合関数のセットを追加。以下のサポートを含む。
    - 1 次スプライン、2 次スプライン、3 次スプライン、ステップワイズ法、ユーザー定義スプライン
    - 最適なパフォーマンスのために構成パラメーターを用いたセル探索
    - ユーザー定義補間 (内挿) および補外 (外挿)
    - ベクトル値関数
    - 列優先および行優先格納形式
- スパース BLAS: インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応のコア数の多いシステムにおいて、非常に疎な行列の CSR (Compressed Sparse Row) 形式の行列-ベクトル乗算 (?CSRMV) のパフォーマンスが向上
- FFT: インテル® AVX 対応システムにおいて、1D 倍精度 FFT のパフォーマンスが向上
- 統計関数: インテル® Core™ プロセッサーにおいて、分散共分散行列および相関行列の計算 (FAST 法) のパフォーマンスとスケーラビリティーが向上
- 問題の修正: http://intel.ly/rITU0r

7.2.10 Update 9 での変更

- LAPACK: 非常に小さなサイズ (~10 x 10) で [C/Z]GEEV のパフォーマンスが向上
- FFT: 大幅なパフォーマンス向上のためにインプレースの 1D FFT 実関数をスレッド化
- FFT: 32 ビットのオペレーティング・システムを実行するインテル® Xeon® プロセッサー E5 ファミリーのシステムにおける 2 の累乗の倍精度複素 1D FFT のスケーラビリティー向上のために新しいアルゴリズムを追加
- 乱数ジェネレーター: 新しいインテル® マイクロアーキテクチャー (開発コード名: Ivy Bridge) ベースのプロセッサーで利用可能なハードウェアをサポートする、RdRand 命令に基づく非決定性乱数ジェネレーターをサポート
- ベクトル算術関数: インテル® Core™ プロセッサーにおいて、Erf() 関数および Pow3o2() 関数のパフォーマンスが向上
- データ適合: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、スプラインベースの評価、微分、積分ルーチンのパフォーマンスが向上
- 問題の修正: http://intel.ly/rITU0r

7.2.11 Update 10 での変更

- BLAS: インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応の 32 ビット・プログラムで dznrm2 および dnrm2 のパフォーマンスが向上
- LAPACK: LAPACK バージョン 3.4.0 のサポートを追加
- データ適合: インテル® Xeon® E7-4870/E5-2690 プロセッサーにおいて、以下の領域における SearchCells1D() 関数のパフォーマンスが向上。
    - 補間点の数が 32 を超える任意の非一様および擬一様領域
    - 補間点の数が 32 未満のすべての種類の領域
- 問題の修正: http://intel.ly/rITU0r

7.3 権利の帰属

エンド・ユーザー・ソフトウェア使用許諾契約書 (End User License Agreement) で言及されているように、製品のドキュメントおよび Web サイトの両方で完全なインテル製品名の表示 (例えば、"インテル® マス・カーネル・ライブラリー") とインテル® MKL ホームページ (http://www.intel.com/software/products/mkl (英語)) へのリンク/URL の提供を正確に行うことが最低限必要です。

インテル® MKL の一部の基となった BLAS の原版は http://www.netlib.org/blas/index.html (英語) から、LAPACK の原版は http://www.netlib.org/lapack/index.html (英語) から入手できます。LAPACK の開発は、E. Anderson、Z. Bai、C. Bischof、S. Blackford、J. Demmel、J. Dongarra、J. Du Croz、A. Greenbaum、S. Hammarling、A. McKenney、D. Sorensen らによって行われました。LAPACK 用 FORTRAN 90/95 インターフェイスは、http://www.netlib.org/lapack95/index.html (英語) にある LAPACK95 パッケージと類似しています。すべてのインターフェイスは、純粋なプロシージャー用に提供されています。

インテル® MKL クラスター・エディションの一部の基となった ScaLAPACK の原版は http://www.netlib.org/scalapack/index.html (英語) から入手できます。ScaLAPACK の開発 は、L. S. Blackford、J. Choi、A. Cleary、E. D'Azevedo、J. Demmel、I. Dhillon、J. Dongarra、S. Hammarling、G. Henry、A. Petitet、K. Stanley、D. Walker、R. C. Whaley らによって行われました。

インテル® MKL の PARDISO は、バーゼル大学 (University of Basel) から無償で提供されている PARDISO 3.2 (http://www.pardiso-project.org (英語)) と互換性があります。

本リリースのインテル® MKL の一部の FFT 関数は、カーネギーメロン大学からライセンスを受けて、SPIRAL ソフトウェア生成システム (http://www.spiral.net/ (英語)) によって生成されました。SPIRAL の開発は、Markus Püschel、José Moura、Jeremy Johnson、David Padua、

Manuela Veloso、Bryan Singer、Jianxin Xiong、Franz Franchetti、Aca Gacic、Yevgen Voronenko、Kang Chen、Robert W. Johnson、Nick Rizzolo らによって行われました。

8 インテル® スレッディング・ビルディング・ブロック

インテル® スレッディング・ビルディング・ブロックの変更に関する詳細は、TBB ドキュメント・ディレクトリーの CHANGES というファイルを参照してください。

9 著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む)に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本書で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、インテルの Web サイトを参照してください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ を参照してください。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。