インテル® C++ コンパイラー 11.1 Windows* 版 プロフェッショナル・エディション インストール・ガイドおよびリリースノート

資料番号: 321414-002JA
2010 年 3 月 24 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。

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。報告されている問題の修正リストは、インテル® コンパイラー 11.1 プロフェッショナル・エディション修正リスト (英語)、インテル® IPP ライブラリー 6.1 修正リスト (英語)、インテル® マス・カーネル・ライブラリー 10.2 修正リスト (英語) およびインテル® スレッディング・ビルディング・ブロック 2.2 の変更を参照してください。

Update 6

- インテル® インテグレーテッド・パフォーマンス・プリミティブが 6.1 Update 5 に更新
- インテル® マス・カーネル・ライブラリーが 10.2 Update 5 に更新
- 報告されている問題の修正

Update 5 (11.1.060)

- インテル® インテグレーテッド・パフォーマンス・プリミティブが 6.1 Update 4 に更新
- インテル® マス・カーネル・ライブラリーが 10.2 Update 4 に更新
- インテル® スレッディング・ビルディング・ブロックが 2.2 Update 2 に更新
- 報告されている問題の修正

Update 4 (11.1.054)

- OpenMP* ヘッダーファイルの変更によりエラー検出が向上
- インテル® インテグレーテッド・パフォーマンス・プリミティブが 6.1 Update 3 に更新
- インテル® マス・カーネル・ライブラリーが 10.2 Update 3 に更新
- 報告されている問題の修正

Update 3 (11.1.051)

- インテル® スレッディング・ビルディング・ブロックが 2.2 Update 1 に更新
- 報告されている問題の修正

Update 2 改訂版 (11.1.048)

- クロスコンパイラー (IA-32 からインテル® 64、IA-32 から IA-64) がリビルド。一部の Windows* 7 システムおよび Windows Server* 2008 システムでコンパイラーが動作しない問題が修正されました。生成されたコードの正当性は問題ではありません。
- Update 2 でインテル® スレッディング・ビルディング・ブロック (インテル® TBB) のバージョンが 2.2 に変更 (以前の注記にはありませんでした)。インテル® 64 アーキテクチャーでは、インテル® TBB のインストール先フォルダーは "em64t" ではなく、"intel64" です。
- サポートされるオペレーティング・システムに Microsoft* Windows 7 が追加

Update 2 (11.1.046)

- 新しいコンパイラー・オプション /Qmkl と /QxAVX に関する注意事項の追加
- 一部のインテル® AVX アーキテクチャー命令と組み込み命令の削除に関する注意事項の追加
- 動作環境の更新。インテル® Parallel Debugger Extension は Microsoft Visual Studio* .NET 2003 でサポートされていない旨の記述が追加されました。
- Visual Basic プロジェクトを含む Visual Studio ソリューションでインテル® C++ 用に変換後、手動で「クリーン」を行う必要がある旨の注意事項を追加
- インテル® 64 アーキテクチャー上のコマンドライン・コンパイルでファイル名に日本語が含まれている場合の診断メッセージの問題に関する注意事項の追加
- インテル® Parallel Debugger Extension におけるタスクウェイトの表示
- 報告されている問題の修正

Update 1 (11.1.038)

- /Od の動作変更に関する注意事項の追加
- MSVCR71.DLL の依存性によりインテル® Parallel Debugger Extension が失敗する問題の修正
- インテル® Parallel Debugger Extension の使用には OpenMP ライブラリーのダイナミック・リンク (デフォルト) の指定が必要である旨の記述の追加
- 報告されている問題の修正

製品リリース (11.1.035)

1.2 製品の内容

インテル® C++ コンパイラー 11.1 Windows 版プロフェッショナル・エディション 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。Windows オペレーティング・システムを実行する IA-32、インテル® 64、および IA-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。
- IA-64 対応アプリケーション開発用インテル® アセンブラー
- 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 6.1 Update 5
- インテル® マス・カーネル・ライブラリー 10.2 Update 5
- IA-32 およびインテル® 64 用インテル® スレッディング・ビルディング・ブロック 2.2 Update 2
- Microsoft Visual Studio 2005/2008 用インテル® Parallel Debugger Extension
- Microsoft 開発環境への統合
- サンプルプログラム
- 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://software.intel.com/en-us/articles/intel-architecture-platform-terminology/ (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) または IA-64 アーキテクチャー (インテル® Itanium® プロセッサー) プロセッサーをベースとするコンピューター
    - 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)
- 4GB のディスク空き容量 (すべての機能およびすべてのアーキテクチャー)
- Microsoft Windows XP、Microsoft Windows Vista*、Microsoft Windows 7、Microsoft Windows Server* 2003、Microsoft Windows Server 2008、Microsoft Windows HPC Server 2008 (エンベデッド・エディションはサポートされていません)
    - Microsoft Windows Server 2008 または Windows HPC Server 2008 では Microsoft Visual Studio 2008 SP1 が必要です。下記にリストされている Visual Studio のその他のバージョンは Windows Server 2008 または Windows HPC Server 2008 ではサポートされていません。
- IA-32 対応アプリケーションまたはインテル® 64 対応アプリケーションのビルドに、Microsoft Visual Studio 開発環境あるいはコマンドライン・ツールを使用する場合は、次のいずれか:
    - Microsoft Visual Studio 2008 Standard Edition または以上の Edition (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
    - Microsoft Visual Studio 2005 Standard Edition または以上の Edition (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
- IA-32 対応アプリケーションのビルドに、Microsoft Visual Studio 開発環境またはコマンドライン・ツールを使用する場合は、次のいずれか:
    - Microsoft Visual Studio .NET 2003 (C++ コンポーネントがインストールされていること) [2]
    - Microsoft Visual C++* .NET 2003 [2]
- IA-64 対応アプリケーションのビルドに、Microsoft Visual Studio 開発環境またはコマンドライン・ツールを使用する場合は、次のいずれか:
    - Microsoft Visual Studio 2008 Team System Edition (C++ コンポーネントと [Itanium コンパイラおよびツール] コンポーネントがインストールされていること) [3] さらに、Windows Server 2008 および .NET Framework 3.5 用 Microsoft Windows SDK
    - Microsoft Visual Studio 2005 Team System Edition (C++ コンポーネントと [Itanium コンパイラおよびツール] コンポーネントがインストールされていること) [3]
- IA-32 アーキテクチャー・アプリケーションのビルドに、コマンドライン・ツールのみを使用する場合は、次のいずれか:
    - Microsoft Visual C++ 2008 Express Edition
    - Microsoft Visual C++ 2005 Express Edition と Windows Server 2008 および .NET Framework 3.5 用 Microsoft Windows SDK
- インテル® 64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合は、次のいずれか:
    - Windows Vista 用 Microsoft Windows Software Development Kit Update
    - Windows Server 2008 および .NET Framework 3.5 用 Microsoft Windows SDK
- IA-64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合:
    - Windows Server 2008 および .NET Framework 3.5 用 Microsoft Windows SDK
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

注:

1. Microsoft Visual Studio 2005/2008 Standard Edition では、[x64 コンパイラおよびツール] コンポーネントがデフォルトでインストールされます。Professional 以上のエディションでは、[カスタム] インストールが必要です。

2. Microsoft Visual Studio .NET 2003 は、Microsoft Windows Vista または Microsoft Windows 7 ではサポートされていません。インテル® Parallel Debugger Extension は Microsoft Visual Studio .NET 2003 ではサポートされていません。本製品の将来のバージョンでは、Microsoft Visual Studio .NET 2003 はサポートされなくなる予定です。
3. IA-64 システムでは、Microsoft Visual Studio はサポートされていません。
4. IA-64 アーキテクチャー・システムでの開発は、IA-64 アーキテクチャー・アプリケーションのビルドのみがサポートされています。
5. 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。
6. アプリケーションは、上記の開発用と同じ Windows バージョンで実行できます。また、Windows XP よりも前の非エンベデッドの Microsoft Windows 32 ビット・バージョンでも実行できますが、インテルではこれらの互換性テストは行われていません。開発アプリケーションが、古いバージョンの Windows にはない Win32 API ルーチンを使用している可能性があります。アプリケーションの互換性テストをご自身の責任で行ってください。アプリケーションを実行するには、特定のランタイム DLL をターゲットシステムにコピーしなければならないことがあります。

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。

1.5 サンプル

製品コンポーネントのサンプルは、「インストール先フォルダー」の説明にある `Samples` フォルダーに用意されています。

1.6 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。

日本語の言語サポート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・システムで英語の言語サポートを使用する場合は、http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-windows/ (英語) の説明を参照してください。

1.7 テクニカルサポート

インストール時にコンパイラーの登録を行わなかった場合は、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

2.1 インストール前の準備

2.1.1 64 ビット・アプリケーション用の Visual Studio の設定

Microsoft Visual Studio 2005 または 2008 を使用し、64 ビット・アプリケーション (インテル® 64 または IA-64 アーキテクチャー向け) を開発する場合は、Visual Studio の構成を変更して、64 ビット・サポートを追加します。

Visual Studio 2005/2008 Standard Edition を使用する場合は、インテル® 64 対応アプリケーションのビルド用に構成を変更する必要はありません。その他のエディションの場合は、次の操作を行ってください。

1. [コントロール パネル] の [プログラムの追加と削除] から [Microsoft Visual Studio 2005 (または 2008)] を選択し、[変更と削除] をクリックします。[Visual Studio メンテナンス モード] ウィンドウが表示されます。[次へ] をクリックします。
2. [機能の追加と削除] をクリックします。
3. [選択した機能をインストールします] で [言語ツール] の [Visual C++] を展開します。
4. [x64 コンパイラおよびツール] ボックスがオンになっていない場合は、オンにし、[更新] をクリックします。ボックスがオンの場合は、[キャンセル] をクリックします。

Microsoft Visual Studio 2005/2008 Team System Edition を使用して、IA-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するには、上記の手順に従い、[Itanium コンパイラおよびツール] ボックスがオンになっていることを確認してください。

2.1.2 Microsoft Windows Vista または Microsoft Windows 7 でのインストール

Microsoft Windows Vista または Microsoft Windows 7 では、Microsoft Visual Studio .NET 2003 はサポートされていません。Microsoft Visual Studio 2005 ユーザーは、*Visual Studio 2005 Service Pack 1* (VS 2005 SP1) と *Visual Studio 2005 Service Pack 1 Update for Windows Vista* (VS 2005 SP1 ページからリンクが提供) をインストールしてください。これらのアップデートをインストールした後に、管理者権限で Visual Studio が実行できることを確認してください。実行できない場合、インテル® コンパイラーを使用できません。詳細は、Microsoft の「Visual Studio on Windows Vista」ページ (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx (英語)) および関連ドキュメントを参照してください。

Microsoft Visual Studio 2008 は、アップデートを適用しなくても Windows Vista および Windows 7 で使用できますが、最新の Microsoft Service Pack を適用することを推奨します。

2.2 インストール

初めて製品をインストールする場合は、インストール中にシリアル番号の入力が求められますので、あらかじめご用意ください。製品のインストールと使用には、有効なライセンスが必要です。

インストールを開始するには、製品 DVD を DVD ドライブに挿入します。自動でインストールが開始されます。自動で開始されない場合は、Windows エクスプローラで DVD ドライブのトップレベル・ディレクトリーを開き、`setup.exe` をダブルクリックします。

製品のダウンロード版を購入した場合は、ダウンロードしたファイル (.EXE) をダブルクリックして、インストールを開始します。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。

2.3 製品の変更、更新、削除

Windows のコントロールパネルの [プログラムの追加と削除] でインストールまたは削除する製品コンポーネントを変更します。

製品のアップデート・バージョンをインストールする際、古いバージョンを最初にアンインストールする必要はありません。複数のバージョンのコンパイラーをインストールし、その中から選択して使用することができます。新しいバージョンのコンパイラーを削除した場合、以前のバージョンの Microsoft Visual Studio への統合を再インストールする必要があります。

2.4 インストール先フォルダー

11.1 製品は、前のバージョンとは異なる構成でフォルダーにインストールされます。新しい構成を以下に示します。一部含まれていないフォルダーもあります。

- C:\Program Files\Intel\Compiler\11.1*xxx*
  - bin
    - ia32
    - ia32_intel64
    - ia32_ia64
    - intel64
    - ia64
  - Documentation
  - help
  - include
  - ipp
    - em64t
    - ia32
    - ia64
  - lib
    - ia32
    - intel64
    - ia64
  - mkl
    - benchmarks
    - em64t
    - examples
    - ia32
    - ia64
    - include
    - interfaces
    - tests
    - tools
  - perf_headers
  - Samples
  - tbb
    - examples
    - ia32
    - include
    - intel64
  - setup_c
  - vsDebuggerExtension

xxx は 3 桁のビルド番号です。bin、include、lib 配下のフォルダーは次のとおりです。

- ia32: IA-32 上で動作するアプリケーションのビルドに使用するファイル
- intel64 または em64t: インテル® 64 上で動作するアプリケーションのビルドに使用するファイル
- ia64: IA-64 上で動作するアプリケーションのビルドに使用するファイル
- ia32_intel64: IA-32 上での実行用のコンパイラー。インテル®64 上で動作するアプリケーションをビルドします。
- ia32_ia64: IA-32 (またはインテル® 64) 上での実行用コンパイラー。IA-64 上で動作するアプリケーションをビルドします。

英語以外の Windows システムにインストールする場合、Program Files フォルダー名が異なる場合があります。インテル® 64 および IA-64 アーキテクチャー・システムでは、フォルダー名は Program Files (X86) またはそれに相当する名前です。

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 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

詳細は、コンパイラーのドキュメントを参照してください。

- C++0x からの追加機能
- C++ ラムダ関数
- 並列実行用の言語拡張
- 非同期 I/O 拡張
- 10 進浮動小数点
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- IPP オプションを使用した valarray の実装
- スレッドデータ共有と再入可能な呼び出し検出に役立つ並列デバッグコードのインストルメンテーション

3.2.1 インテル® C++ プロジェクト・ファイルの互換性

インテル® C++ プロジェクト・ファイル (.icproj) の形式がバージョン 11.0 で変更されました。インテル® C++ の古いバージョンで作成されたプロジェクトを開くと、プロジェクトの変換が必要である旨のメッセージが表示されます。バージョン 11 のプロジェクトを古いバージョンのインテル® C++ 統合で使用することはできません (ただし、古いバージョンのコンパイラーは、[ツール] > [オプション] > [Intel C++ (インテル(R) C++)] - [Compilers (コンパイラー)] から使用できます)。

3.3 新規および変更されたコンパイラー・オプション

詳細は、コンパイラーのドキュメントを参照してください。

- /hotpatch
- /Qdiag-enable:sc-parallel
- /Qmkl[:lib]
- /QxAVX
- /Yd

3.3.1 /Od オプションの /Op の除外

バージョン 11.1 では、最適化を無効にする /Od オプションは、浮動小数点精度を最大にする /Op を含意しなくなりました。/Op スイッチは廃止予定です。そのため、浮動小数点精度の影響を受けやすいアプリケーションには、明示的に /fp オプションを指定することを推奨します。

3.3.2 廃止予定のオプション

/YX オプション (プリコンパイル済みヘッダーを自動で生成) は廃止予定で、インテル® C++ コンパイラーの将来のメジャーリリースでは削除されます。Microsoft では、このオプションを Microsoft Visual Studio 2005 から削除しています。代わりに /Yu、/Yc を使用してください。

その他の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 その他の変更および既知の問題

3.4.1 最適化レポートがデフォルトで無効に設定

バージョン 11.1 以降、コンパイラーは、ベクトル化、自動並列化、OpenMP スレッド化ループに関する最適化レポートメッセージをデフォルトで表示しないようになりました。これらのメッセージを表示するには、`/Qdiag-enable:vec`、`/Qdiag-enable:par`、`/Qdiag-enable:openmp` を指定するか、`/Qvec-report`、`/Qpar-report`、`/Qopenmp-report` を使用する必要があります。

また、バージョン 11.1 以降、最適化レポートメッセージは `stdout` ではなく、`stderr` に送られます。

3.4.2 ビルド環境コマンドスクリプトの変更

ビルド環境を構築するコマンド・ウィンドウ・スクリプトが使用する Microsoft Visual Studio バージョンを任意で指定できるよう 11.1 で変更されました。ビルド環境ウィンドウを開くのに、定義済みのスタート・メニュー・ショートカットを使用していない場合は、次のコマンドを使用して適切な環境を構築してください。

```
"C:\Program Files\Intel\Compiler\11.1\xxx\Bin\iclvars.bat" arch [vs]
```

xxx は、リビジョン番号です。*arch* は、`ia32`、`ia32_intel64`、`intel64`、`ia32_ia64`、`ia64` のいずれかです (「インストール先フォルダー」を参照)。*vs* は任意で指定します。`vs2008` または `vs2005` のいずれかです。*vs* が指定されていない場合は、コマンドライン統合用にインストール時に指定された Visual Studio のバージョンがデフォルトで使用されます。Microsoft Visual Studio .NET 2003 は *vs* 引数で指定できないことに注意してください。

コンパイラーを異なるパスにインストールしている場合は、適切なフォルダーを指定しください。

3.4.3 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

バージョン 11.0 以降、IA-32 アーキテクチャー向けのコンパイルでは、`/arch:SSE2` (旧: `/QxW`) がデフォルトになりました。`/arch:SSE2` 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。プログラムが命令をサポートしていないプロセッサーで実行された場合は、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、`/arch:IA32` を指定してください。

3.4.4 OpenMP ライブラリーのデフォルトが "compat" に変更

バージョン 10.1 では、新しい OpenMP ライブラリー・セットが追加され、アプリケーションは、インテル® コンパイラーと Microsoft コンパイラーの両方からの OpenMP コードを使用することが可能でした。この "互換" ライブラリーは古い "レガシー" ライブラリーよりも高いパフォーマンスを提供します。バージョン 11.0 以降、互換ライブラリーが OpenMP アプリケーションのデフォルト・ライブラリーとして使用されるようになりました。`/Qopenmp-lib:compat` と等価です。古いライブラリーを使用する場合は、`/Qopenmp-lib:legacy` を指定してください。

"レガシー" ライブラリー (libguide.lib、libguide40.lib など) は、インテル® コンパイラーの将来のリリースからは削除される予定です。

3.4.5 サンプルの提供 (ZIP 形式)

バージョン 11.1 以降、コンパイラー・プログラミング・サンプルは zip アーカイブで提供されます。各 ZIP アーカイブを書き込み可能なフォルダーに展開してください。すべてのサンプルは Visual Studio ソリューションとして提供され、コマンドライン・ビルドの説明も提供されます。詳細は、`samples.htm` ファイルを参照してください。

3.4.6 Visual Studio .NET 2003 のドキュメント表示エラー

Microsoft Visual Studio .NET 2003 で Microsoft MSDN ライブラリー機能をインストールしていない場合、*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・ヘルプ* または F1 状況依存ヘルプのメニュー項目を使用すると、「Visual Studio 用にヘルプがインストールされていません」というメッセージが表示されます。この問題を回避するには、[ヘルプ] > [目次] をクリックした後に表示されるエラーメッセージで [OK] をクリックしてください。[目次] ペインから製品のヘルプにアクセスできるようになります。この問題は、Microsoft Visual Studio 2005 または 2008 では発生しません。

3.4.7 バージョン管理システムでのインテル® C++ プロジェクトの使用

プロジェクトがバージョン管理システム (例: Microsoft Visual SourceSafe* や Microsoft Visual Studio Team Foundation Server など) で管理されている場合、プロジェクトでインテル® C++ プロジェクト・システムを使用するには追加のステップが必要です。このトピックについての詳細な記事は、http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/ (英語) を参照してください。

3.4.8 インテル® C++ 用に変換後のプロジェクトをクリーンアップする新しいオプション

プロジェクトでインテル® C++ を使用するためにこのオプションが選択されると、新しいダイアログボックスが表示され、プロジェクトの「クリーン」機能が提供されます (以前のビルド結果を削除します)。これは推奨されるステップで、デフォルトで選択されています。「クリーン」機能を無効にして手動でクリーンアップを行うこともできます。

ただし、英語以外の Microsoft Visual Studio や Microsoft Visual Basic* プロジェクトを含むソリューションを使用している場合は、このダイアログは表示されません。手動でプロジェクトを「クリーン」にする必要があります。

3.4.9 日本語ファイル名に関するコマンドライン診断表示の問題

コンパイル診断で日本語が含まれているファイル名は、ネイティブのインテル® 64 対応アプリケーション用コンパイラーを使用して、Windows コマンドでコンパイルした場合に正しく表示されません。Visual Studio を使用する場合やインテル® 64 対応アプリケーション用クロスコンパイラーまたは IA-32 対応アプリケーション用コンパイラーを使用する場合は、この問題は発生しません。

3.4.10 一部のインテル® AVX アーキテクチャー命令と組み込み命令の削除

VPERMIL2PD 命令と VPERMIL2PS 命令はサポートされていません。また、対応する組み込み命令 _mm256_permute2_pd、_mm_permute2_pd、_mm256_permute2_ps、_mm_permute2_ps もサポートされていません。これらの命令や組み込み命令はインテル® AVX アーキテクチャーから削除されていますが、コンパイラー・ドキュメントからはその記述が削除されていません。詳細は、http://software.intel.com/en-us/blogs/2009/01/29/recent-intelr-avx-architectural-changes/ (英語) を参照してください。

3.4.11 OpenMP ヘッダーファイルの変更

バージョン 11.1 Update 4 では、OpenMP ヘッダーファイル omp.h にエラーチェック機能が追加されました。

omp_lock_t 型と omp_nest_lock_t 型の定義が変更されています。本リリースでは、コンパイラーはこれらの型をコンパイル時に識別します。この変更は、OpenMP 仕様に準拠した OpenMP プログラムには影響しません。OpenMP 仕様に準拠していない OpenMP プログラムでは、コンパイラーが警告を発行することがあります。次に例を示します。

```
> type sample.c
#include <omp.h>
int main() {
    omp_lock_t lk;
    omp_init_nest_lock( & lk );
    return 0;
} // main

> icl /Qopenmp sample.c

sample.c(4): warning #167: argument of type "omp_lock_t *" is
incompatible with parameter of type "omp_nest_lock_t *"

      omp_init_nest_lock( & lk );
                          ^
```

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

このセクションでは、インテル® C++ コンパイラー・プロフェッショナル・エディションに同梱されているインテル® 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://software.intel.com/en-us/intel-ipp/ (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://software.intel.com/en-us/intel-ipp/ (英語)) のドキュメントのリンクを参照してください。

4.1 新機能と変更された機能

4.1.1 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 6.1 Update 5

- 本リリースでは、新機能は追加されていません。報告されている問題が修正されています。

4.1.2 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 6.1 Update 4

- 『インテル® IPP 信号処理リファレンス・マニュアル』にストリング処理に関する新しいコード例を追加
- ライブラリーに RSA-1024 暗号化の最適化を追加
- OpenSSL パフォーマンスの向上と OpenSSL. 0.9.8j のサポート

4.1.3 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 6.1 Update 3

- 『インテル® IPP 信号処理リファレンス・マニュアル』の第 11 章に新しいコード例を追加
- UMC ドキュメントに動き検出とモード決定のセクションを追加
- BZIP2 デコーダーでパフォーマンスが約 5% 向上

4.1.4 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 6.1 Update 2

- インテル® Advanced Vector Extensions (インテル® AVX) のサポート
- インテル® Core™ i7 プロセッサーの新しい最適化とスレッド化制御/最適化をサポート
- 3D 画像処理: 3D 幾何学変換、3D フィルター
- 新しいデータ圧縮関数 API
- RSA_SSA1.5 と RSA_PKCSv1.5 の新しいインテル® IPP 暗号化サポート
- PNG 形式サポートを追加する UIC (Unified Image Classes) と DXT1、DXT3、DXT5 画像圧縮をサポートする新しい機能
- 球面調和関数とパーリンノイズ生成関数を含む高度な光関数
- Windows Media* Photo (HD Photo) のサポート: IPP PCT 関数
- MPEG-2 のシーン解析、VC1 の輝度補償とオーバーラップ・スムージングを含む新しいビデオ・コーディング分野の向上
- 信号処理、画像処理、ストリング処理、C++/C# 言語サポートのサンプルを \Samples フォルダーに追加。その他のサンプルは、http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-intel-ipp-intel-ipp-sample-code/ (英語) からダウンロードできます。
- 廃止予定の API のさらに多くのリファンレス情報がリファレンス・マニュアルとヘッダーファイルに追加

4.2 既知の制限事項

- 一部の「生成ドメイン」関数 ("ippg" 接頭辞の関数) で Visual Studio の状況依存ヘルプが動作しないことがあります。
- 問題の修正リスト、既知の問題、制限事項については、次のナレッジベースの記事を参照してください。http://software.intel.com/en-us/articles/intel-ipp-library-61-fixes-list/ (英語)

4.3 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://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/ (英語) を参照してください。

4.4 インテル® IPP コードサンプル

インテル® IPP コードサンプルとして、Windows 版、Linux 版、Mac OS 版のダウンロード・パッケージが用意されています。以下の Web サイトから入手できます。
http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/ (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。

5 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® C++ コンパイラー・プロフェッショナル・エディションに同梱されているインテル® 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。

5.1 本バージョンでの変更

本バージョンおよび以前のバージョンの変更についての詳細は、
http://software.intel.com/en-us/articles/new-in-intel-mkl-10-2/ (英語) を参照してください。

5.1.1 インテル® マス・カーネル・ライブラリー 10.2 Update 5

新機能

- LAPACK 3.2.1 アップデート (主に LAPACK 3.2 に関する修正) に対応

パフォーマンスの向上

- FFT
  - インテル® 64 アーキテクチャー上で 3 次元以上の複素数 FFT のパフォーマンスが向上
- VSL
  - 45nm インテル® Core™2 Duo プロセッサー以降の 64 ビット・ライブラリーで、MT19937 と MT2203 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上

ユーザービリティーとインターフェイスの向上

- uBLAS の例で Boost 1.41.0 のサポートを追加
- 対角優位ソルバー関数 (?DTSVB、?DTTRFB、?DTTRSB) に Fortran 95 インターフェイスを追加
- インプレース多次元クラスター FFT のメモリー消費を大幅に削減

5.1.2 インテル® マス・カーネル・ライブラリー 10.2 Update 4

新機能

- 単精度複素数の絶対値を求める SCABS1 関数を追加
- 部分的なピボット演算を使用する一般的なソルバー (?GTSV) よりも 2 倍高速な対角優位の三重対角方程式用の ?DTSVB ソルバーを追加
- 対角優位の三重対角方程式用の因数分解ルーチン (?DTTRFB) と 前方/後方代入ルーチン (?DTTRSB) を追加

パフォーマンスの向上

- FFT
  - 8 または 13 の倍数の変換のパフォーマンスが向上
  - ベクトル長が 2 のべき乗でない 1D 複素数クラスター FFT を最適化
- VSL
  - デシメーションを必要とする畳み込み/相関演算のパフォーマンスが大幅に向上 (再リンクが必要。「既知の問題」を参照)

5.1.3 インテル® マス・カーネル・ライブラリー 10.2 Update 3

パフォーマンスの向上

- BLAS
  - インテル® Core™ i7 プロセッサーおよびインテル® Xeon® プロセッサー 5300 番台、5400 番台、5500 番台で次の BLAS レベル 1、2 関数の 32 ビット OS バージョンがスレッド化: (D,S,C,Z)COPY、(D,S,C,Z)SWAP、(D,S,C,Z)AXPY、(S,C)ROT、(S,C)DOT、CDOTC、(D,S,C,Z)GEMV、(D,S,C,Z)TRMV、(S,C)SYMV、(S,C)SYR、(S,C)SYR2
  - インテル® Xeon® プロセッサー 5300、5400、5500 で次の BLAS レベル 1 関数の 32 ビットおよび 64 ビット OS バージョンが向上: ZAXPY、ZSCAL、ZDOT(U,C)、(D,S)ROT
  - インテル® Xeon® プロセッサー 5300 で列より非常に多い行を持つ行列の DGEMM スレッド化効率が向上
- LAPACK
  - 次の LAPACK 関数のスケーラビリティーが向上: ?POTRF、?GEBRD、?SYTRD、?HETRD、?STEDC 分割統治固有ソルバー
- FFT
  - 下層のカーネルが更新され、FFT において幅広くパフォーマンスが向上
  - 1 つの関数呼び出しで少数の変換が計算される場合に 3D FFT のスレッド化が向上
  - スレッド化を小規模の多次元変換に拡張
- VML
  - インテル® Xeon® プロセッサー 5500 番台の VML 関数がさらに最適化: v(s,d)Asin、v(s,d)Acos、v(s,d)Ln、v(s,d)Log10、vsLog1p、v[s/d]Hypot
- VSL
  - viRngPoisson および viRngPoissonV 乱数ジェネレーターのパフォーマンスが向上

ユーザービリティーとインターフェイスの向上

- uBLAS、Java、FFTW3、LAPACK95、BLAS95 のサンプルプログラムの向上
- examples ディレクトリーとリファレンス・マニュアルのソースコードにある同一のサンプルは、リファレンス・マニュアルでは削除
- クラスター FFT の新しい 64 ビット整数 (ILP64) fftw_mpi インターフェイス

5.1.4 インテル® マス・カーネル・ライブラリー 10.2 Update 2

新機能

- LAPACK 3.2
  - 238 個の新しい LAPACK 関数
  - 超精密反復法の改良
  - ハウスホルダー QR 因数分解の非負対角
  - 低プロファイル行列でのハイパフォーマンス QR とハウスホルダー反射
  - 高速で正確な新しいヤコビ法 SVD
  - 矩形フル圧縮形式のルーチン
  - ピボットコレスキー
  - 混合精度反復法の改良 (コレスキー)
  - より安定した DQDS アルゴリズム
- DZGEMM 拡張 BLAS 関数の実装 (http://www.netlib.org/blas/blast-forum/ (英語) の説明を参照)。リファレンス・マニュアルの BLAS セクションの *gemm 関数ファミリーの説明を参照してください。
- PARDISO で実数、複素数、単精度データをサポート

ユーザービリティー/インターフェイスの向上

- スパース行列形式変換ルーチン:
  - CSR (3-配列バリエーション) ↔ CSC (3-配列バリエーション)
  - CSR (3-配列バリエーション) ↔ 対角形式
  - CSR (3-配列バリエーション) ↔ スカイライン
- Fortran95 BLAS と LAPACK のコンパイル・モジュール・ファイル (.mod) を追加
  - モジュールは、インテル® Fortran コンパイラーで事前にビルドされており、インクルード・ディレクトリーにあります (フルパス情報については、インテル® MKL ユーザーズ・ガイドを参照してください)。
  - ほかのコンパイラー用のソースも提供されています。
  - インターフェイスについてのドキュメントは、インテル® MKL ユーザーズ・ガイドを参照してください。
- FFTW3 インターフェイスを直接メイン・ライブラリーに統合
  - デフォルトのインテル® Fortran コンパイラー規則と名前修飾で互換性のないコンパイラーでラッパーを作成するためのソースコードも提供されています。
  - 詳細は、リファレンス・マニュアルの付録 G を参照してください。
- DFTI_DESCRIPTOR_HANDLE を型の名前に追加。ユーザープログラムで型として参照できます。
- ユーザーデータへのアクセスを可能にするために最適化ソルバードメインのヤコビ行列計算ルーチンにパラメーターを追加 (詳細は、リファレンス・マニュアルの djacobix 関数の説明を参照してください)
- 64 ビット・アーキテクチャーでインテル® MKL の単精度 BLAS 関数 (頭文字 "s" または "c" の関数) から 64 ビット浮動小数点精度関数へのインターフェイス・マッピング呼び出しを追加 (詳細は、インテル® MKL ユーザーズ・ガイドの「sp2dp」を参照してください)
- 互換ライブラリー (「ダミーライブラリー」) を削除

パフォーマンスの向上

- インテル® 64 アーキテクチャー用にさらにスレッド化された BLAS レベル 1、2 関数
  - レベル 1 関数 (ベクトル-ベクトル): (CS,ZD,S,D)ROT、(C,Z,S,D)COPY、(C,Z,S,D)SWAP
    - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.7 ～ 4.7 倍向上
    - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 14 ～ 130 倍向上
  - レベル 2 関数 (行列-ベクトル): (C,Z,S,D)TRMV、(S,D)SYMV、(S,D)SYR、(S,D)SYR2
    - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.9 ～ 2.9 倍向上
    - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 16 ～ 40 倍向上
- インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー (5300 番台、5400 番台、7400 番台) で、DSYRK の 32 ビット逐次バージョンに導入された再帰アルゴリズムのパフォーマンスが最大 20% 向上。
- インテル® Xeon® 7460 プロセッサーで、大規模な問題の LU 因数分解 (DGETRF) がバージョン 10.1 Update 1 に対して 25% 向上。また小規模な問題でも劇的に向上。
- BLAS *TBMV/*TBSV 関数でレベル 1 BLAS 関数を使用。インテル® Core™ i7 プロセッサー上で 最大 3%、インテル® Core™2 プロセッサー 5300 番台と 5400 番台で最大 10% のパフォーマンスが向上。
- DGEMM パフォーマンスを強化するスレッド化アルゴリズムの向上
  - 8 スレッドで最大 7% の向上、3、5、7 スレッドで最大 50% の向上 (インテル® Core™ i7 プロセッサー)
  - 3 スレッドで最大 50% の向上 (インテル® Xeon® プロセッサー 7400 番台)
- 非素数サイズのスレッド化 1D 複素数-複素数 FFT
- 3D 複素数-複素数変換の新しいアルゴリズムにより 1 スレッドまたは 2 スレッドで小さな問題サイズ (最大 64x64x64) についてより優れたパフォーマンスを提供
- 対称正定行列の演算時におけるアウトオブコア (OOC) PARDISO のハイレベルな並列化実装
- すべての行列の型でインコアとアウトオブコアの両方で PARDISO のメモリー使用量が減少
- 実対称行列、複素エルミート行列、複素対称行列に対し PARDISO OOC で使用されるメモリーがインテル® MKL 10.1 で使用されていたメモリーの半分以下まで減少
- PARDISO/DSS における順序付けの並列化とシンボリック因子分解
- インテル® Core® i7 プロセッサーとインテル® Core™2 プロセッサーで次の VML 関数において最大 2 倍のパフォーマンスの向上 (平均で 30% の向上): v(s,d)Round、v(s,d)Inv、v(s,d)Div、v(s,d)Sqrt、v(s,d)Exp、v(s,d)Ln、v(s,d)Atan、v(s,d)Atan2
- インテル® Advanced Vector Extension (インテル® AVX) で次の関数の最適化バージョンが利用可能
  - BLAS: DGEMM
  - FFT
  - VML: exp、log、pow
  - 上記の関数にアクセスする `mkl_enable_instructions()` 関数に関する重要な情報については、インテル® MKL ユーザーズ・ガイドを参照してください。

5.2 既知の問題

本リリースにおける既知の制限事項の詳細なリストは、http://software.intel.com/en-us/articles/intel-math-kernel-library-support-resources/ (英語) を参照してください。

5.3 注意事項

インテル® MKL の将来のバージョンでは以下の変更が予定されています。「テクニカルサポート」を参照してください。

- ファイル名に `solver` を含むライブラリーの内容をコア・ライブラリーに移動する予定です。これらの `solver` ライブラリーはその後削除される予定です。

5.4 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 (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。本リリースのインテル® MKL の一部の FFT 関数は、ヒューストン大学からライセンスを受けて、UHFFT ソフトウェア生成システム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。

6 インテル® スレッディング・ビルディング・ブロック

このセクションでは、インテル® C++ コンパイラー・プロフェッショナル・エディションに同梱されているインテル® スレッディング・ビルディング・ブロック (インテル® TBB) 2.2 の変更点、新機能、および最新情報をまとめています。

6.1 本バージョンでの変更

6.1.1 インテル® スレッディング・ビルディング・ブロック 2.2 Update 2

- parallel_invoke と parallel_for_each で関数オブジェクトを値ではなく定数参照として利用可能
- /MT オプションでのインテル® TBB のビルドをサポート。Visual C++ ランタイム DLL の特定のバージョンへの依存を回避できます。/MT を指定してビルドした DLL は、vc_mt ディレクトリーに生成されます。
- critical_section クラス の追加
- 例外サポートの向上: 新しい例外クラスが追加され、すべての例外はアウトオブライン内部メソッドでスローされます。
- TBB のアロケーターと malloc の置換に関する修正と向上 (安定したメモリー識別、Windows 上でのダイナミック関数置換を含む)
- tbb_thread クラスに swap() メソッドを追加
- concurrent_hash_map に rehash() メソッドと bucket_count() メソッドを追加
- コード、テスト、サンプル、ドキュメントでのその他の修正と追加/変更

6.1.2 インテル® スレッディング・ビルディング・ブロック 2.2 Update 1

- ドキュメントのアップデート
- TBB スケジューラーの自動初期化が可能性のあるすべての使用事例に対応
- concurrent_queue: パディングで使用される sizeof の引数の型が実際に使用される型と一致
- メモリー・アロケーターの向上: scalable_malloc を呼び出す可能性のある malloc をサポート (Windows 以外)。TBB メモリー・アロケーター起動時のメモリー割り当て要求の処理を修正。
- Windows の malloc の置換でスタティック・オブジェクトのサポートが向上
- 並列処理できないパイプラインの起動時に、シングルスレッドでの実行を保証、アイドルスピンを排除、パフォーマンスが向上
- RML のリファクタリングとクリーンアップ
- 複数のアイテムの領域を予約するための concurrent_hash_map 用の新しいコンストラクターを追加
- TBB の例外クラスに delete() 演算子を追加
- parallel_reduce でのラムダサポートの強化
- concurrent_queue の gcc 4.3 警告の修正
- グローバル・スタティック・オブジェクトの構築中、TBB エンティティーを使用するモジュールで初期化時に発生する可能性のあるデッドロックを修正
- concurrent_hash_map のコピー・コンストラクターの修正
- 特定の場合にスケジューラーでクラッシュする問題の修正
- メモリー不足の場合に TBB アロケーターでクラッシュする問題の修正
- ラムダ式がサポートされているかどうかをチェックする test_lambda.cpp を含む新しいテストの実装
- concurrent_vector の安全性問題に関する既知の例外の修正
- concurrent_vector における同時拡張要求の並列処理が向上
- TBB アロケーターで大きなオブジェクトの割り当てのパフォーマンスがさらに向上
- Linux 上でのテキストの再配置ソースの問題を修正
- Windows 上での malloc の置換に関する問題を修正
- コード、テスト、サンプル、ドキュメントでのその他の小さな変更

6.2 既知の問題

インテル® スレッディング・ビルディング・ブロックの本リリースに関する次の注意事項に留意してください。

6.2.1 複数の Visual Studio バージョンがインストールされている場合の問題

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® TBB) は、Microsoft Visual Studio の各種バージョンをサポートする DLL セットを個別に提供しています。サポートされている複数の Visual Studio バージョンがインストールされている場合 (例: 2005 と 2008)、インテル® スレッディング・ビルディング・ブロックをインストールするとサポートされているすべての Visual Studio バージョンに対し、システムの PATH 環境変数に DLL フォルダーが追加されます (通常は最新バージョンが最初にリストされます)。Visual Studio のバージョンが異なっても DLL は同じ名前のため、PATH で見つかった最初のセットのみが使用されます。これにより、1 つの Visual Studio バージョンでビルドされ、異なるバージョンの DLL を使用して実行されるアプリケーションで問題が発生する可能性があります。

複数の Visual Studio バージョンがインストールされている場合、インテル® TBB を使用したアプリケーション開発では、1 つのバージョンを選択して、その他のバージョンのフォルダーを PATH から削除することを推奨します。フォルダー名は以下の形式です。

- VS2003: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc7.1\bin
- VS2005: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc8\bin
- VS2008: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc9\bin

フォルダーパスは、アーキテクチャーやシステム言語により異なります。インテル® スレッディング・ビルディング・ブロックはスタティック・ライブラリーを提供していません。

6.2.2 ライブラリーの問題

- インテル® スレッド・チェッカーまたはインテル® スレッド・プロファイラーを使用した際により正確な結果を得るには、インテル® TBB とともに使用する前にそれらの製品の最新のアップデート・リリースをダウンロードしてください。
- 同じプログラムで連続してインテル® TBB と OpenMP コンストラクトをともに使用していて、OpenMP コードにインテル® コンパイラーを使用している場合、KMP_BLOCKTIME に小さな値 (例えば、20 ミリ秒) を設定するとパフォーマンスが向上します。この設定は、kmp_set_blocktime() ライブラリー呼び出しを使用して OpenMP コード内で行うこともできます。KMP_BLOCKTIME および kmp_set_blocktime() の詳細は、コンパイラーの OpenMP に関するドキュメントを参照してください。
- 一般に、アプリケーションやサンプルの非デバッグ (“リリース”) ビルドは、インテル® TBB ライブラリーの非デバッグバージョンとリンクし、デバッグビルドはインテル® TBB ライブラリーのデバッグバージョンとリンクします。Windows システムでは、/MD オプションを使用してコンパイルした場合はインテル® TBB ライブラリーのリリース・ライブラリー、/MDd オプションを使用してコンパイルした場合はデバッグ・ライブラリーを使ってビルドしてください。他の組み合わせでは、ランタイムエラーが発生します。デバッグ・ライブラリーとリリース・ライブラリーの詳細については、製品の “Documentation” サブディレクトリーに含まれているチュートリアルを参照してください。

7 インテル® Parallel Debugger Extension

このセクションでは、インテル® C++ コンパイラー・プロフェッショナル・エディションのインテル® Parallel Debugger Extension の変更点、新機能、および最新情報をまとめています。

7.1 既知の問題

- Microsoft Visual Studio 2005 を使用している場合、3 つのインテル固有の例外を手動で有効に設定する必要があります。[デバッグ] > [例外] を選択し、[Win32 Exceptions] ツリーを展開して、以下の項目を有効にします。

  ```
  a1a01db0 Intel Parallel Debugger Extension Exception 0
  a1a01db1 Intel Parallel Debugger Extension Exception 1
  a1a01db2 Intel Parallel Debugger Extension Exception 2
  ```

  これは、プロジェクトごとに 1 回設定します。

- デバッグセッション中にインテルのデバッグ例外を無効にすると、Visual Studio (Visual Studio 2008 SP1 まで) がハングアップすることがあります。
- インテル® Parallel Debugger Extension を使用するには、OpenMP ライブラリーが動的にリンクされている必要があります (デフォルト)。インテル® Parallel Debugger Extension を使用する場合、OpenMP ライブラリーのスタティック・リンクを指定する /Qopenmp-link:static を使用しないでください。
- 並列デバッグを行う前に並列デバッグ・インストルメンテーションを有効にしてください ([プロパティ]> [構成プロパティ] > [C/C++] > [Debug (デバッグ)] > [Enable Parallel Debug Checks (並列デバッグ検証を有効にする)])。この設定を行わない場合、デバッガーはデータ共有イベントや再入可能な呼び出しでの中断を検出できません。
- Microsoft Visual Studio 2008 を使用し、64 ビット・アプリケーションのデバッグを行う場合、Visual Studio 2008 Service Pack 1 がインストールされている必要があります。
  - サービスパックがインストールされていない場合、Visual Studio 2005 および 2008 での 64 ビット・アプリケーションのデバッグは、低メモリー領域にリンクされる場合のみ行うことができます。低メモリー領域にリンクされない場合、デバッグ対象が終了するまでイベントは表示されません。終了後、すべてのイベントがイベントウィンドウに表示されます。64 ビット・アプリケーションを適切にデバッグするには、[プロジェクト] > [プロパティ] > [Linker (リンカー)] > [Advanced (詳細)] でベースアドレスを 0x10000 に設定します。
- [Data Sharing Events (データ共有イベント)] ウィンドウでローカル変数が「???」と表示されます。
- SSE レジスターウィンドウが 64 ビット・アプリケーションで動作しません。ウィンドウに「???」と表示されます。
- スタティック・ローカル変数のフィルターがコンテキスト・メニューから正しく設定されません。
- 逆アセンブルビューで再入可能な呼び出しの検出が停止します。スタティック関数の場合、正しく動作しません。デザインモードでは、{,,myapp.exe} my_extern_function など、適切なコンテキスト演算子の後で関数を使用してください。
- デバッガー拡張ウィンドウの配置が "docked" から "floating" に変更されるとウィンドウは空のままです。この問題を回避するには、"docked" のままにしておくか、または配置の変更後にデバッグセッションを再起動します。
- デバッガー拡張では、Visual Studio からアプリケーションを開始する必要があります。既存のプロセスへアタッチしている場合は動作しません。
- ウィンドウが非表示、あるいは閉じられた後に再度開かれた場合は、デフォルト (16 進) 設定に戻ります。

7.2 ドキュメント

インテル® Parallel Debugger Extension のドキュメントは、Microsoft Visual Studio の [ヘルプ] メニュー、または特定のダイアログボックスで [ヘルプ] ボタンをクリックして表示することができます。状況依存ヘルプも [Debugger Extension (デバッガー拡張)] ウィンドウをアクティベートした後、F1 キーをクリックして表示できます。

8 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。

MPEG-1、MPEG-2、MPEG-4、H.263、H.264、MP3、DV SD/25/50/100、VC-1、G.722.1、G.723.1A、G.726、G.728、G.729、GSM/AMR、GSM/FR、JPEG、JPEG 2000、Aurora、TwinVQ、AC3 および AAC は、ISO、IEC、ITU、SMPTE、ETSI およびその他の組織によって制定されている国際標準規格です。これらの標準規格の実装、または標準規格対応のプラットフォームの使用には、インテルを含むさまざまな組織からのライセンス許諾が必要になる場合があります。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国および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。