インテル® C++ コンパイラー 16.0 Update 1 for Windows* リリースノート (インテル® Parallel Studio XE 2016)

2015 年 12 月 7 日時点

このドキュメントは、インテル® デベロッパー・ゾーンに公開されている「Intel C++ Compiler 16.0 Update 1 for Windows* Release Notes for Intel Parallel Studio XE 2016」の日本語参考訳です。

このドキュメントでは、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。

詳細は、パッケージに含まれるライセンスと本リリースノートの「著作権と商標について」を参照してください。本リリースのインテル® C++ コンパイラー 16.0 Update 1 についての詳細は、次のリンクを参照してください。

変更履歴

Update 1 (インテル® C++ コンパイラー 16.0.1)

- OpenMP* 4.1 ドラフト仕様 TR3 の新機能をサポート
- インテル® SIMD Data Layout Templates (インテル® SDLT)
- OpenMP* オフロードコンパイル (/Qopenmp-offload[:mic|gfx|host]) をサポート
- 報告された問題を修正
- ドキュメントを更新

インテル® C++ コンパイラー 15.0 以降 (インテル® C++ コンパイラー 16.0.0 での変更)

- OpenMP* 4.0 の機能を追加サポート
- 10 進浮動小数点を C++ Windows* でサポート
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けの新しい targetptr および preallocated オフロード修飾子
- 複数の同時処理を 1 つの CPU スレッドからインテル® MIC アーキテクチャーへオフロードする新しいオフロードストリームをサポート
- インテル® グラフィックス・テクノロジーへのオフロードのサポートを検証する gfx_sys_check ユーティリティー
- インテル® グラフィックス・テクノロジーの共有ローカルメモリーをサポート
- #pragma simd で cilk_for ループを利用可能に
- 式の演算子の順序を決定するときに括弧を考慮可能に
- C++14 の機能をサポート
- C11 の機能をサポート
- コンパイラーの組込み関数を内部的に定義
- BLOCK_LOOP および NOBLOCK_LOOP プラグマ、unroll_and_jam プラグマの private 節を追加
- 新規および変更されたコンパイラー・オプション
- インテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーをサポート
- ランタイムにインテル® グラフィックス・テクノロジー機能の有無を問い合わせる API を追加
- ランタイムにスレッド空間構成を設定する API を追加

動作環境

- 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)
- RAM 2GB (4GB 推奨)
- 4GB 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)
- インテル® MIC アーキテクチャー向けの開発/テスト:
    - インテル® Xeon Phi™ コプロセッサー
    - インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS)
    - オフロードコードのデバッグには Microsoft* Visual Studio* 2012 以降が必要
- インテル® グラフィックス・テクノロジーへのオフロードまたはネイティブサポートの開発/テスト
    - 次のプロセッサー・モデルをサポートしています。
        - インテル® Xeon® プロセッサー E3-1285 v3 製品ファミリーおよび E3-1285L v3 製品ファミリー (インテル® C226 チップセット) (インテル® HD グラフィックス P4700)

        - 第 5 世代インテル® Core™ プロセッサー (インテル® Iris™ Pro グラフィックス、インテル® HD グラフィックス)
        - 第 4 世代インテル® Core™ プロセッサー (インテル® Iris™ Pro グラフィックス、インテル® Iris™ グラフィックス、またはインテル® HD グラフィックス 4200+ シリーズ)
        - 第 3 世代インテル® Core™ プロセッサー (インテル® HD グラフィックス 4000/2500)

          注: リストされているチップセットのインテル® Xeon® プロセッサーのみサポートしています。ほかのチップセットのインテル® Xeon® プロセッサーはサポートしていません。前世代のインテル® Core™ プロセッサーはサポートしていません。インテル® Celeron® プロセッサーおよびインテル® Atom™ プロセッサーとの互換性はありません。
    - インテル® グラフィックス・テクノロジー対応の最新の 32 ビットまたは 64 ビット・グラフィックス・ドライバー (インテル® ダウンロード・センター (英語) から入手できます)
    - Windows* 用 binutils (http://intel.ly/1fHX7xO (英語) から入手できます)
        - binutils をインストールした後、ld.exe を含むディレクトリーを PATH に追加する必要があります。
- Microsoft* Windows* 7、Microsoft* Windows* 8、Microsoft* Windows* 8.1、Microsoft* Windows* 10、Microsoft* Windows* 2008 SP2 (IA-32 のみ)、Microsoft* Windows Server* 2008 (R2 SP1)、Microsoft* Windows* HPC Server 2008、Microsoft* Windows Server* 2012 (エンベデッド・エディションはサポートしていません) [3]
    - Microsoft* Windows* 8、Microsoft* Windows* 8.1 および Microsoft* Windows Server* 2012 では、製品は「デスクトップ」環境にインストールされます。「Windows* 8 UI」アプリケーションの開発はサポートされていません。[4]
- IA-32 対応アプリケーション [2] 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* 2015 Professional Edition 以上 (「Visual C++ 2015 用の共通ツール」コンポーネントがインストールされていること) [5]
    - Microsoft* Visual Studio* Community 2015 以上 (「Visual C++ 2015 用の共通ツール」コンポーネントがインストールされていること) [5]
    - Microsoft* Visual Studio* 2013 Professional Edition 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* Community 2013 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* 2012 Professional Edition 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* 2010 Professional Edition 以上 (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 [2] 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++* Express 2013 for Windows Desktop
  - Microsoft* Visual C++* Express 2012 for Windows Desktop
  - Microsoft* Visual C++* 2010 Express Edition
- インテル® 64 対応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++* Express 2013 for Windows Desktop
  - Microsoft* Visual C++* Express 2012 for Windows Desktop
  - Microsoft* Windows* Software Development Kit for Windows* 8
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

注

1. Microsoft* Visual Studio* 2010 では、このコンポーネントがデフォルトで含まれています。
2. 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。
3. 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。
4. インテル® C++ コンパイラーは、Windows 8* UI アプリの開発をサポートしていません。インテルでは、ユーザーの皆様のご意見を常に参考にしています。例えば、Windows* 8 UI アプリケーションにインテル® C++ コンパイラーまたはその他のインテル® ソフトウェア開発製品の機能を利用したい方は、インテル® プレミアサポート (https://premier.intel.com/ (英語)) からご意見をお送りください。
5. インテル® C++ コンパイラーを Microsoft* Visual Studio* 2015 で使用するには、Visual Studio* から「Visual C++ 2015 用の共通ツール」コンポーネントをインストールする必要があります。この記事 (英語) の説明を参照してください。

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS)

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) は、インテル® C++ コンパイラーのインストール前またはインストール後にインストールできます。

最新バージョンのインテル® MPSS を使用することを推奨します。インテル® Parallel Studio XE for Windows* を登録すると、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 (http://registrationcenter.intel.com) から入手できます。

ユーザー空間およびカーネルドライバーのインストールに必要な手順については、インテル® MPSS のドキュメントを参照してください。

インテル® C++ コンパイラーの使用方法

コマンドラインおよび Microsoft* Visual Studio* からのインテル® C++ コンパイラーの使用方法についての情報は、『入門ガイド』 (*<install-dir>\documentation_2016\ps2016\getstart_comp_wc.htm*) に含まれています。

ドキュメント

製品ドキュメントは、*<install-dir>\documentation_2016\ja\ps2016\getstart_comp_wc.htm* からリンクされています。

Microsoft* Visual Studio* のオンラインヘルプ形式

オンラインヘルプ形式はブラウザーベースです。Microsoft* Visual Studio* の [ヘルプ] メニューからインテルのドキュメントを参照する場合、または *F1* キー、ダイアログボックスにあるヘルプボタン、その他の GUI で状況依存ヘルプを参照する場合、デフォルトのブラウザーに対応するヘルプトピックが表示されます。デフォルトのブラウザーによっては、いくつかの小さな問題が発生することがあります。次のような既知の問題があります。

- [ヘルプ設定の設定] が [ブラウザーで起動] に設定されている場合、[ツール] > [オプション] > [F# ツール] または [ツール] > [オプション] > [Intellitrace] で *F1* キーを押すと、ブラウザーが 2 つ開きます。
- **Chrome*:** 検索またはキーワードからトピックを表示すると、目次が同期しません。[トピックを同期] も動作しません。
- **Firefox*:** 目次が表示されなくなることがあります。検索の大文字と小文字は区別されます。
- **Safari*:** Windows* の反応が遅くなります。

Windows Server* 2012 の Microsoft* Internet Explorer* 10 でドキュメントが表示されない問題

Windows Server* 2012 の Internet Explorer* 10 でヘルプまたはドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、信頼済みサイトのリストに "about:internet" を追加します。オプションで、ドキュメントを参照した後に信頼済みサイトのリストから "about:internet" を削除できます。

Windows Server* 2012 で Visual Studio* 2012 のドキュメントを表示できない場合

Windows Server* 2012 で Visual Studio* 2012 のヘルプまたはドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、[インターネット] ゾーンで [MIME スニッフィングを有効にする] および [アクティブ スクリプト] を有効にします。

インテルが提供するデバッグ・ソリューション

- インテルが提供するデバッグ・ソリューションは GNU* GDB ベースです。詳細は、「インテル® Parallel Studio 2016 Composer Edition C++ - デバッグ・ソリューション リリースノート」 (英語) を参照してください。

サンプル

製品サンプルは、*<install-dir>\samples_2016\ja\compiler_c\psxe* ディレクトリーにあります。

テクニカルサポート

インストール時に製品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 (http://registrationcenter.intel.com) 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: 販売代理店がこの製品のテクニカルサポートを提供している場合、インテルではなく販売代理店にお問い合わせください。

新機能と変更された機能

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。

OpenMP* 4.1 ドラフト仕様 TR3 の新機能をサポート

インテル® コンパイラー 16.0 における OpenMP* 4.1 ドラフト仕様 TR3 の新しい機能のサポートは、OpenMP* 4.5 仕様 (2015 年 11 月にリリース予定) の規格に合わせて変更される可能性があります。

- *#pragma omp simd simdlen(n)* をサポート
- *#pragma omp ordered simd* をサポート
- *processor* 節の拡張を *#pragma omp declare simd* に追加
- *#pragma omp declare simd* ディレクティブの *linear* 節を新しい修飾子で拡張
    - *linear (linear-list [: linear-step])* (*linear-list* は次のいずれか)
        - *list*
        - *modifier (list)* (*modifier* は *ref*、*val*、*uval* のいずれか)
    - すべての *list* 項目は各 SIMD レーンで呼び出される関数の引数でなければなりません。
    - *modifier* が指定されない場合や *val* または *uval modifier* が指定された場合、各レーンの各 *list* 項目の値は、関数に入るときの *list* 項目の値とレーンの論理番号の倍数 *linear-step* に相当します。
    - *uval modifier* が指定された場合、各呼び出しは各 SIMD レーンと同じメモリー位置を使用します。このメモリー位置は論理的な最終レーンの最後の値で更新されます。
    - *ref modifier* が指定された場合、各レーンの各 *list* 項目のメモリー位置は、レーンの論理番号の倍数 *linear-step* でインデックスされた関数に入るときのメモリー位置の配列に相当します。

インテル® SIMD Data Layout Templates (インテル® SDLT)

- インテル® SDLT は、SIMD ベクトル化の熟練者に手を借りることなく、SIMD ハードウェアとコンパイラーを最大限に活用できるよう支援するライブラリーです。
- インテル® SDLT は、ISO C++11、インテル® Cilk™ Plus SIMD 拡張、および *#pragma ivdep* をサポートしているすべてのコンパイラーで使用できます。
- インテル® SIMD Data Layout Templates の特長
    - ベクトル化のデータレイアウトが SOA の場合でも AOS 形式でプログラム可能
    - パフォーマンス効率の良い SIMD ベクトル化をターゲット
    - ほかの明示的な SIMD プログラミング・モデルと互換
- インテル® SDLT で効率的な SIMD コードを生成するには、次のコンパイラー・オプションが必要です (または推奨します)。
    - C++11 のサポート
        - */Qstd:c11* (Windows*) または *-std=c11* (Linux*)
    - レベル 2 以上のコード最適化
        - */O2* および */O3* (Windows*) または *-O2* および *-O3* (Linux*)
    - 最適化における ANSI エイリアシング規則
        - /Qansi-alias (Windows*) または -ansi-alias (Linux*)
    - ターゲットにするプロセッサー機能をコンパイラーにオプションで指示します。
        - /Qx*code* (Windows*) または -x*code* (Linux*)
    - パフォーマンス上の利点がある場合、複数の機能固有の自動ディスパッチ・コードを生成するようにコンパイラーにオプションで指示します。

- C++11 依存関係により、インテル® SDLT を Linux* で使用する場合は GCC 4.7 以降が必要です。インテル® SDLT を STL アルゴリズムで使用する場合は GCC 4.8 以降を推奨します。
- C++11 依存関係により、インテル® SDLT を Windows* で使用する場合は Microsoft* Visual Studio* 2012 以降が必要です。

OpenMP* オフロードコンパイル (/Qopenmp-offload[:mic|gfx|host]) をサポート

- Visual Studio* IDE 統合のプロパティー「OpenMP* オフロードコンパイル」と「OpenMP* オフロードコンパイルのターゲットデバイス」が組み合わされ、1 つのプロパティー 「OpenMP* オフロードコンパイルの有効化とターゲットデバイスの指定」になりました。この新しいオプションは、ターゲット *pragma* の OpenMP* オフロードコンパイルおよびターゲットデバイスの指定を有効/無効にします。デフォルトのターゲットデバイスは ***mic*** です。プロパティー「OpenMP* オフロードコンパイルのターゲットデバイス」は古いオプション (非推奨) で、将来のリリースで削除される予定です。

OpenMP* 4.0 の機能を追加サポート

- ユーザー定義リダクションを定義する *#pragma omp declare reduction* をサポート
- *#pragma omp simd collapse(n)* をサポート

10 進浮動小数点を C++ Windows* でサポート

Linux* (C/C++ の両方) および Windows* (C のみ) で、IEEE 規格 754-2008 で定義された 10 進浮動小数点をサポートしました。C++ のサポートは、ISO/IEC TR 24733:2011 の記述に従って追加されました。

インテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーをサポート

インテル® C++ コンパイラーでインテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーのサポートを追加しました。これらのライブラリーは、/Qmkl および /Qtbb (Windows*) または -mkl および -tbb (Linux* / OS X*) の両方のオプションを指定し、/Qopenmp 、-qopenmp、-fopenmp オプションを指定しない場合に使用されます。OpenMP* が必要なときにインテル® TBB を使用するインテル® MKL ライブラリーをリンクする場合は、リンク時にライブラリーを明示的に指定する必要があります。

ランタイムにインテル® グラフィックス・テクノロジー機能の有無を問い合わせる API を追加

ソースレベルでランタイムにインテル® グラフィックス・テクノロジーに関するハードウェア情報を取得する API *_GFX_get_device_platform(void)*、*_GFX_get_device_sku(void)*、*_GFX_get_device_hardware_thread_count(void)*、*_GFX_get_device_min_frequency(void)*、*_GFX_get_device_mas_frequency(void)*、および *_GFX_get_device_current_frequency(void)* が追加されました。

ランタイムにスレッド空間構成を設定する API を追加

アプリケーションでランタイムにスレッド空間とスレッドグループの高さと幅を制御できる API *_GFX_set_thread_space_config(int, int, int, int)* が追加されました。アプリケーションでランタイムにスレッド空間の高さと幅を制御できる GFX_THREAD_SPACE_WIDTH および GFX_THREAD_SPACE_HEIGHT 環境変数が追加されました。スレッドグループの高さと幅を制御できる GFX_THREAD_GROUP_WIDTH および GFX_THREAD_GROUP_HEIGHT 環境変数もあります。

インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けの新しい targetptr および preallocated オフロード修飾子

オフロード構文に 2 つの新しい修飾子 *targetptr* および *preallocated* が追加されました。これらの修飾子を使用すると、*#pragma offload (targetptr)* から、またはオフロードコード (*preallocated targetptr*) により、インテル® MIC アーキテクチャー専用のメモリーを割り当てることができます。

複数の同時処理を 1 つの CPU スレッドからインテル® MIC アーキテクチャーへオフロードする新しいオフロードストリームをサポート

ターゲットデバイスで並列に実行される、インテル® MIC アーキテクチャーへの複数処理のオフロードを実装する、新しい API (*_Offload_stream_handle*、*_Offload_stream_completed*、*_Offload_device_streams_completed*) および追加のオフロードプラグマの節 (*stream*、*signal*) が利用できるようになりました。オフロードストリーム API の詳細は、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』を参照してください。

オフロードストリームを使用するには、次の環境変数を設定する必要があります。

- MIC_ENV_PREFIX=MIC
- MIC_KMP_AFFINITY=norespect,none
- OFFLOAD_STREAM_AFFINITY={compact | scatter}

インテル® グラフィックス・テクノロジーへのオフロードのサポートを検証する gfx_sys_check ユーティリティー

プラットフォームがインテル® グラフィックス・テクノロジーへのオフロードをサポートしているか確認するユーティリティー *gfx_sys_check* が提供されました。このユーティリティーは、インテル® グラフィックス・テクノロジーに関連する詳細も提供します。

インテル® グラフィックス・テクノロジーの共有ローカルメモリーをサポート

インテル® グラフィックス・テクノロジーのスレッド間で共有ローカルメモリーを有効にして、RAM トラフィックを減らしパフォーマンスを向上する、新しい *cilk_for* ループ *_Thread_group* 節、新しい *__thread_group_local* 変数属性、新しい API *_gfx_gpgpu_thread_barrier()* が提供されました。

#pragma simd で cilk_for ループを利用可能に

インテル® Cilk™ Plus の *cilk_for* キーワードを使用して並列化された C/C++ for ループを、*#pragma simd* または *_Simd* キーワードを使用して、明示的なベクトル化のターゲットにできるようになりました。

式の演算子の順序を決定するときに括弧を考慮可能に

/Qprotect-parens (Windows*) および *-fprotect-parens* (Linux*/OS X*) オプションを指定すると、コンパイラーは式を評価するときに括弧を考慮し、演算の順序を変更しません。例えば、これらのオプションを使用すると、コンパイラーは次の変換を行いません。

double y = (a + b) + c;
から
double y = a + (b + c);

これらのオプションは、デフォルトでは無効です。これらのオプションを使用すると、パフォーマンスに悪影響を及ぼすことがあります。

C++14 の機能をサポート

インテル® C++ コンパイラー 16.0 は、*/Qstd:c++14* (Windows*) または *-std=c++14* (Linux*/OS X*) コンパイラー・オプションで C++14 の機能をサポートします。

- 以前の主要バージョンのコンパイラーとの比較を含む、サポートしている機能の最新リストは、「インテル® C++ コンパイラーでサポートされる C++14 の機能」を参照してください。

C11 の機能をサポート

インテル® C++ コンパイラー 16.0 は、*/Qstd:c11* (Windows*) または *-std=c11* (Linux*/OS X*) コンパイラー・オプションで C11 の機能をサポートします。

- 以前の主要バージョンのコンパイラーとの比較を含む、サポートしている機能の最新リストは、「インテル® C++ コンパイラーにおける C11 サポート」を参照してください。

コンパイラーの組込み関数を内部的に定義

コンパイル時間を短縮するため、インテル® C++ コンパイラー 16.0 では組込み関数宣言のヘッダーファイルをチェックしないようになりました。この変更が型チェックに影響する可能性があるため、関数プロトタイプを追加する、*-D__INTEL_COMPILER_USE_INTRINSIC_PROTOTYPES* コンパイラー・オプションが追加されました。

BLOCK_LOOP および NOBLOCK_LOOP プラグマ、unroll_and_jam プラグマの private 節を追加

ループ・ブロッキング情報をコンパイラーに伝える新しいプラグマと更新されたプラグマが追加されました。これには、新しいプラグマ *#pragma BLOCK_LOOP [clause[[,] clause...]]* および *#pragma NOBLOCK_LOOP* も含まれます。さらに、*#pragma unroll_and_jam* への *private(var list)* 節の追加も含まれます。詳細は、『インテル® C++ コンパイラー・ユーザー・リファレンス・ガイド』を参照してください。

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・オプションの詳細は、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』の「コンパイラー・オプション」セクションを参照してください。

- /Qdaal[: lib] インテル® Data Analytics Acceleration Library (インテル® DAAL) の特定のライブラリーにリンクするようにコンパイラーに指示します。
- /Qoffload-svm 共有仮想メモリー (SVM) モードを使用するかどうかを指定します。このオプションは、インテル® グラフィックス・テクノロジーにのみ適用されます。
- /Qopt-prefetch-issue-excl-hint インテル® マイクロアーキテクチャー Broadwell (開発コード名) 以降で prefetchW 命令をサポートします。

廃止予定のコンパイラー・オプションのリストは、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』の「コンパイラー・オプション」セクションを参照してください。

終了予定のサポート

Microsoft* Visual Studio* 2010 のサポートを終了予定

Microsoft* Visual Studio* 2010 のサポートは、将来のリリースで終了する予定です。

IA-32 ホスト・インストールのサポートを終了予定

IA-32 ホストへのインストールのサポートは、将来のリリースで終了する予定です。

終了したサポート

スタティック解析のサポートを終了

スタティック解析のサポートを終了しました。ご意見やお問い合わせは、こちら (英語) までお寄せください。

Microsoft* Visual Studio* 2008 のサポートを終了

Microsoft* Visual Studio* 2008 のサポートを終了しました。

既知の制限事項

ポインターチェッカーにダイナミック・ランタイム・ライブラリーが必要

/Qcheck-pointers オプションを使用する場合は、ランタイム・ライブラリー *libchkp.dll* をリンクする必要があります。*/MD* のようなオプションを */Qcheck-pointers* とともに使用すると、設定に関係なくこのダイナミック・ライブラリーがリンクされることに注意してください。詳細は、http://intel.ly/1jV0eWD (英語) を参照してください。

日本語版 Windows* にインストールすると、IDE からインテル® コンパイラーのヘルプ・ドキュメントを起動できない

インテル® Parallel Studio XE 2016 を日本語版 Windows* にインストールすると、Microsoft* Visual Studio* IDE からインテル® コンパイラーのヘルプ・ドキュメントを起動できないことがあります。この問題の詳細は、こちら (英語) を参照してください。

異なるコンパイラーを使用して 256 ベクトル・ビット型引数をコンパイルするとランタイムにアクセス違反が発生するコードが生成される

2 つの異なるコンパイラー (Microsoft* Visual C++* 2013 コンパイラーおよびインテル® C++ コンパイラー15.0 以降) を使用してアプリケーションを作成した場合、アライメントされていないデータアクセスによる一般保護違反が発生することがあります。この問題は、256 ベクトル・ビット型引数が呼び出しの際に参照で渡され、呼び出し元を Visual C++* でビルドし、その引数をインテル® C++ コンパイラーでビルドした関数でアクセスすると発生します。

原因は、256 ベクトル・ビット型引数のアライメントが一致しないためです。

<code> に AVX 以降の新しいコード値 (CORE-AVX-I、 CORE-AVX2、その他) を指定して */Qx<code>* コンパイラー・オプションを使用した場合は、アプリケーションのソースコードで *__mm256_stream_** (非テンポラルデータのロード/ストア組込み関数) が明示的に使用されない限り、アライメントされていないアクセス命令がこれらのインスタンスで実際に使用されるため、この問題は発生しません。

Visual Studio* の既知の問題

- 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

  Microsoft* Visual Studio* 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://intel.ly/pImnpo (英語) を参照してください。

- *MSVCP90D.dll* (またはその他の Microsoft* ランタイム DLL) が見つからない

  サンプル・プロジェクト (および Microsoft* Visual C++* プロジェクト) を実行するときに Microsoft* Visual Studio* のランタイム DLL が見つからない場合、ランタイムエラーが発生します。これは、マニフェスト・ファイルや SXS アセンブリーが見つからないことが原因です。この問題を解決するには、使用しているバージョンの Microsoft* Visual Studio* の redist フォルダー (デフォルトの場所は *c:\program files[(x86)]\Microsoft Visual Studio X.X\VC\redist*) に移動します。*amd64*、*x86*、*Debug_NonRedist* サブフォルダーで、必要なランタイムが含まれているフォルダーを探します (デバッグ・ライブラリーを探す場合は、ファイル名の最後が D のファイルが含まれているフォルダーを探します)。必要なランタイムが含まれているフォルダーが見つかったら、そのフォルダーの (.manifest ファイルを含む) すべての内容を、実行する .exe ファイルのあるフォルダーにコピーします。

- Visual Studio* 2010 では */fp:precise* がデフォルトでオン

  Visual Studio* 2010 で作成または変換されたプロジェクトでは、デフォルトで */fp:precise* コマンドライン・オプションがオンになります。このオプションは、パフォーマンスを低下させるいくつかの最適化を無効にして、浮動小数点演算の一貫性を向上させる「浮動小数点モデル」を設定します。インテルのデフォルトである */fp:fast* に戻すには、プロジェクトのプロパティー・ページで [C/C++] > [Code Generation (コード生成)] > [Floating Point Model (浮動小数点モデル)] を *Fast* に変更します。

- Visual Studio* 2010 の言語パック

  インテル® C++ コンパイラーをインストールした後に Visual Studio* 2010 の新しい言語パックをインストールすると、[プロジェクト プロパティ] ダイアログにインテル® C++ コンパイラー固有のオプションが表示されなくなることがあります。その場合には、以下の手順を試してみてください。

1. "*<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\1033*" ディレクトリーが存在する場合は、すべてのファイルを "*<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\<locale-ID>*" にコピーします。
2. "*<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033*" が存在する場合は、すべてのファイルを "*<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>*" にコピーします。

   * *<locale-ID>* は言語パックを表します。

   別の方法として、インテル® C++ コンパイラーをアンインストールして、再インストールすることもできます。

- Microsoft* Edge* の状況依存 (F1) ヘルプ問題

- Microsoft* Edge* を Microsoft* Visual Studio* のデフォルトブラウザーとして設定している場合、特定の機能/関数で状況依存 (F1) ヘルプを呼び出すと、機能/関数の説明に関連するトピックの代わりに、対応するドキュメントのタイトルページが表示されます。正しい動作にするには、異なるデフォルトブラウザーを使用するように Microsoft* Visual Studio* の設定を変更してください。

インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーの既知の問題

- _Cilk_shared の制限
  - 仮想基本クラスで _Cilk_shared 属性を指定することはできません。
  - 複数の基本クラスから _Cilk_shared 属性が指定されたクラスを派生させることはできません (複数の継承は許可されません)。
  - _Cilk_shared 属性が指定されたクラスで仮想デストラクターを定義することはできません。
  - _Cilk_shared 属性が指定されたクラスを別の _Cilk_shared クラスの基本クラスとして使用する場合、そのサイズが 8 の倍数になるように (必要に応じて、仮のフィールドを追加して) プログラマーが調整する必要があります。
  - _Cilk_offload は、共有ライブラリー (DLL) を使うプログラムでは使用できません
- 共有ライブラリーに含まれるコードをオフロードする際に */Qoffload=mandatory* オプションまたは */Qoffload=optional* オプションを指定してメインプログラムのリンクが必要

  オフロードには初期化処理が必要ですが、これはメインプログラムでのみ行うことができます。つまり、共有ライブラリーに含まれるコードをオフロードする場合、初期化処理が行われるように、メインプログラムもリンクしなければなりません。メインコードやメインプログラムへスタティック・リンクされたコードにオフロード構造が含まれる場合、これは自動で行われます。そうでない場合、*/Qoffload=mandatory* コンパイラー・オプションまたは */Qoffload=optional* コンパイラー・オプションを指定して、メインプログラムをリンクする必要があります。

- オフロード・コンパイル・モデルでリンク時に見つからないシンボルの検出

  リンク時に見つからないシンボルを検出するために */Qoffload-option,mic,compiler,"-z defs"* を指定する必要はなくなりました。コンパイラー・ドライバーが、この検出を自動的に行います。

- コンパイル時の診断の **MIC** タグ

  ターゲット (インテル® MIC アーキテクチャー) とホスト CPU のコンパイルを区別できるようにコンパイラーの診断インフラストラクチャーが変更され、出力メッセージに **MIC** タグが追加されました。このタグは、インテル® MIC

アーキテクチャー用のオフロード拡張を使用してコンパイルしたときに、ターゲットのコンパイル診断にのみ追加されます。

下記の例で、サンプル・ソース・プログラムは、ホスト CPU とターゲット (インテル® MIC アーキテクチャー) のコンパイルの両方で同じ診断を行っています。ただし、プログラムによっては、2 つのコンパイルで異なる診断メッセージが出力されます。新しいタグが追加されたことで、CPU とターゲットのコンパイルを容易に区別できることが分かります。

$ icl -c sample.c

*sample.c(1): 警告 #1079: *MIC* 関数 "main" の戻り型は "int" でなければなりません。*
void main()
^
*sample.c(5): 警告 #120: *MIC* 戻り値の型が関数の型と一致しません。*
return 0;
^
sample.c(1): 警告 #1079: 関数 "main" の戻り型は "int" でなければなりません。
void main()
^
sample.c(5): 警告 #120: 戻り値の型が関数の型と一致しません。
return 0;

- ランタイム型情報 (RTTI) は未サポート

  仮想共有メモリー・プログラミングでは、ランタイム型情報 (RTTI) はサポートされていません。特に、*dynamic_cast<>* と *typeid()* の使用はサポートされていません。

- 直接 (ネイティブ) モードにおけるランタイム・ライブラリーのコプロセッサーへの転送

  インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) に、*/lib* 以下のインテル® コンパイラーのランタイム・ライブラリー (例えば、OpenMP* ライブラリー *libiomp5.so*) が含まれなくなりました。

  このため、直接モード (例えば、コプロセッサー・カード上) で OpenMP* アプリケーションを実行する場合は、アプリケーションを実行する前にインテル® MIC アーキテクチャー OpenMP* ライブラリー (*<install_dir>\compilers_and_libraries_2016\windows\lib\mic\libiomp5.so*) のコピーをカード (デバイス名の形式は micN。最初のカードは *mic0*、2 番目のカードは *mic1*、...) に (*scp* 経由で) アップロードする必要があります。

  このライブラリーが利用できない場合、次のようなランタイムエラーが発生します。

/libexec/ld-elf.so.1: "sample" で要求された共有オブジェクト "libiomp5.so" が見つかりません。

libimf.so のような別のコンパイラー・ランタイムでも同様です。必要なライブラリーは、アプリケーションおよびビルド構成により異なります。

- オフロード領域からの *exit()* の呼び出し

  オフロード領域から *exit()* を呼び出すと、"オフロードエラー: デバイス 0 のプロセスがコード 0 で予想外に終了しました" のような診断メッセージが出力され、アプリケーションが終了します。

インテル® グラフィックス・テクノロジーへのオフロードの既知の問題

- gfx_linker: : error : command 'ld.exe' exited with non-zero exit code - 107374170

  x64 プロジェクトでインテル® グラフィックス・テクノロジーへコードをオフロードすると、binutils に含まれている *ld.exe* でリンカーエラーが表示されることがあります。この問題を解決するには、64 ビット用の binutils bin ディレクトリーを *PATH* 環境変数に追加して、Microsoft* Visual Studio* を再起動してください。

- オフロードコードのホストバージョンが並列化されない

  コンパイラーは、#pragma offload 以下に並列ループのターゲットバージョンとホストバージョンの両方を生成します。ホストバージョンは、オフロードが実行できない場合 (通常は、ターゲットシステムにインテル® グラフィックス・テクノロジーが有効なユニットがない場合) に実行されます。並列ループは、オフロードの並列セマンティクスを含む cilk_for の並列構文または配列表記文を使用して指定する必要があります。ターゲットバージョンはターゲット実行の際に並列化されますが、現在、ホスト側のバックアップ・バージョンが並列化されない制限があります。cilk_for を使用したときにオフロード実行が行われないと、バックアップ・コード実行のパフォーマンスに大きく影響する場合があることに注意してください。配列表記文は現在ホスト側で並列コードを生成しないため、パフォーマンスに影響はありません。これは既知の問題で、将来のリリースで修正される予定です。

- インテル® グラフィックス・テクノロジーへのオフロードの既知の制限事項
  - Windows* 7 では、オフロードが行われたときにディスプレイをロックできません。アクティブ・ディスプレイが必要です。
  - オフロードコードでは、次の機能を使用できません。
    - 例外処理
    - RTTI
    - *longjmp/setjmp*
    - VLA
    - 変数引数リスト

        - 仮想関数、関数ポインター、その他の間接呼び出しまたはジャンプ
        - 共有仮想メモリー
        - 配列や構造体のようなポインターを含むデータ構造
        - ポインターまたは参照型のグローバル変数
        - OpenMP*
        - *cilk_spawn* または *cilk_sync*
        - インテル® Cilk™ Plus のレデューサー
        - ANSI C ランタイム・ライブラリー呼び出し (SVML、*math.h*、*mathimf.h* 呼び出し、およびその他いくつかの例外あり)
    - 64 ビット浮動小数点演算および整数演算は非効率

インテル® Cilk™ Plus の既知の問題

- スチールが行われた後、対応する *_Cilk_sync* の前に SEH 例外がスローされると、Microsoft* C++ 構造化例外処理 (SEH) は失敗します。

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 (*/Qipo*) 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。

著作権と商標について

最適化に関する注意事項
インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。 注意事項の改訂 #20110804

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場合があります。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。

本資料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。

本資料で紹介されている資料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 (http://www.intel.com/design/literature.htm (英語)) 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。

インテル® C++ コンパイラーは、インテルのソフトウェア使用許諾契約書 (EULA) の下で提供されます。

詳細は、製品に含まれるライセンスを確認してください。

Intel、インテル、Intel ロゴ、Celeron、Cilk、Intel Atom、Intel Core、Intel Xeon Phi、Iris、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。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。