IBM XL C/C++ for Linux, V13.1

はじめに

バージョン 13.1

SA88-5392-00
(英文原典：SC27-4247-00)

IBM XL C/C++ for Linux, V13.1

はじめに

バージョン 13.1

SA88-5392-00
(英文原典：SC27-4247-00)

お願い

本書および本書で紹介する製品をご使用になる前に、 67 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM XL C/C++ for Linux, V13.1 (プログラム 5765-J08; 5725-C73)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。正しいレベルの製品をご使用になるようお確かめください。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC27-4247-00
IBM XL C/C++ for Linux, V13.1
Getting Started with XL C/C++
Version 13.1
First edition

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

本書には、IBM® XL C/C++ for Linux, V13.1 コンパイラーの概要および基本的な使用法に関する情報が含まれています。

本書の対象読者

本書は、XL C/C++ の基本的な概要および使用法に関する情報を必要とする C および C++ の開発者向けです。 コマンド行コンパイラー、 C および C++ プログラミング言語の基本的な知識、およびオペレーティング・システム・コマンドの基本的な知識に習熟していることを前提としています。 XL C/C++ に慣れていないプログラマーは、本書を使用して XL C/C++ 固有の能力や機能に関する情報を確認することができます。

本書の読み方

特に記載がない限り、この解説書の本文はすべて、C と C++ 言語の両方に関連しています。言語間に差異が存在する場合は、『規則』 での説明のように、修飾付きテキストやアイコンで示しています。

本書では、コンパイラーの動作の説明には **xlc** および **xlc++** コンパイラー呼び出しを使用しています。ただし、特定の環境では必要に応じてコンパイラー呼び出しコマンドの別の形式に置き換えることができます。また、コンパイラー・オプションの使用法は、特に指定がない限り同じです。

本書は、コンパイラー環境の構成、XL C/C++ コンパイラーを使用した C または C++ アプリケーションのコンパイルおよびリンクに関する情報をカバーしていますが、以下のトピックは含まれていません。

- コンパイラー・インストール: XL C/C++ のインストールの詳細については、「*XL C/C++ インストール・ガイド*」を参照してください。
- コンパイラー・オプション: コンパイラー・オプションの構文および使用法について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。
- C または C++ プログラミング言語: 構文、セマンティクス、および IBM による C または C++プログラミング言語の実装については、「*XL C/C++ ランゲージ・リファレンス*」を参照してください。
- プログラミング・トピック: プログラムの移植性および最適化に焦点を置いた、XL C/C++ でのアプリケーションの開発について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」を参照してください。

規則

活字の規則

以下の表では、IBM XL C/C++ for Linux, V13.1 の資料で使用されている活字の規則について説明します。

表 1. 活字の規則

書体	意味	例
太字	小文字のコマンド、実行可能ファイル名、コンパイラー・オプション、およびディレクティブ。	コンパイラーには、さまざまな C/C++ 言語レベルおよびコンパイル環境をサポートするために、**xlc** と **xlC** (**xlc++**) という基本呼び出しコマンドとその他のいくつかのコンパイラー呼び出しコマンドが備わっています。
イタリック	パラメーターまたは変数。実際の名前と値はユーザーによって提供されます。イタリックは新規用語の導入にも使用されます。	要求された *size* よりも大きいものを戻す場合には、 *size* パラメーターの更新を確認してください。
下線	コンパイラー・オプションまたはディレクティブのパラメーターのデフォルト設定。	nomaf | maf
モノスペース	プログラミング・キーワードおよびライブラリー関数、コンパイラー・ビルトイン、プログラム・コードの例、コマンド・ストリング、またはユーザー定義の名前。	myprogram.c をコンパイルおよび最適化するには、`xlc myprogram.c -O3` と入力します。

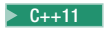
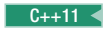
限定を示すエレメント (アイコン)

本書に記述されているフィーチャーの大半は、C と C++ 言語の両方に適用されます。あるフィーチャーが 1 つの言語に限定される場合、あるいは言語間で機能が異なる場合の言語エレメントの説明では、以下のように、アイコンを使用してテキストのセグメントを説明します。

表 2. 限定を示すエレメント

修飾子/アイコン	意味
C のみ、または C のみの始まり   C のみの終わり	このテキストは C 言語のみでサポートされているフィーチャーを記述しています。または、C 言語に特定の振る舞いを記述しています。
C++ のみ、または C++ のみの始まり   C++ のみの終わり	このテキストは C ++ 言語のみでサポートされているフィーチャーを記述しています。または、C++ 言語に特定の振る舞いを記述しています。

表 2. 限定を示すエレメント (続き)

修飾子/アイコン	意味
IBM の拡張機能、または IBM の拡張機能の始まり   IBM の拡張機能の終わり	テキストは、標準の言語仕様に対する IBM 拡張機能であるフィーチャーを説明します。
C11、または C11 の始まり   C11 の終わり	このテキストは、C11 の一部として標準 C に導入されるフィーチャーを記述しています。
C++11、または C++11 の始まり   C++11 の終わり	このテキストは、C++11 の一部として標準 C++ に導入されるフィーチャーを記述しています。

構文図

本書中では、ダイアグラムは XL C/C++ 構文を図示します。このセクションは、これらのダイアグラムの解釈と使用に役立ちます。

- 構文図は線のパスに沿って、左から右、上から下へと読んでいきます。

  ▶▶── 記号は、コマンド、ディレクティブ、またはステートメントの開始を示します。

  ──▶ 記号は、コマンド、ディレクティブ、またはステートメント構文が次の行に続いていることを示します。

  ▶── 記号は、コマンド、ディレクティブ、またはステートメントが前の行から続いていることを示します。

  ──▶◀ 記号は、コマンド、ディレクティブ、またはステートメントの終了を示します。

  完結したコマンド、ディレクティブ、またはステートメント以外の構文単位の図であるフラグメントは、|── 記号で始まり ──| 記号で終わります。
- 必須項目は、次のように横線 (メインパス) 上に表示されます。

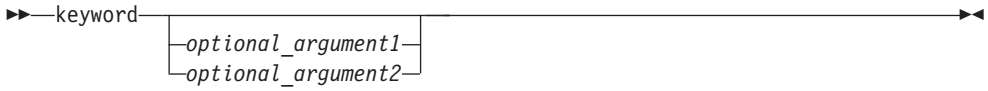
```
▶▶─keyword─required_argument─────────────────────────────────────────▶◀
```

- オプショナル項目は、次のようにメインパスの下側に表示されます。

- 2 つ以上の項目から選択できる場合は、縦に重ねて表示されます。

  項目の中から 1 つを*選択しなければならない* 場合は、スタックの 1 つの項目がメインパスに表示されます。

  項目の 1 つを選択することがオプショナルの場合は、スタック全体がメインパスの下に表示されます。



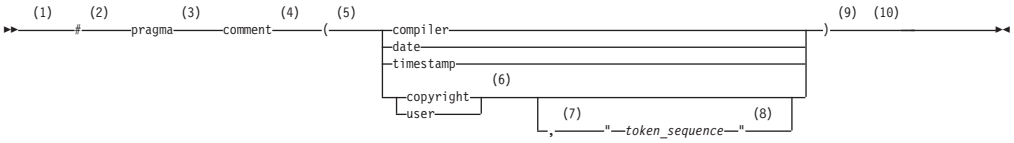
- 主線の上にある左に戻る矢印 (反復矢印) は、スタックされた項目から複数個選択できること、あるいは単一の項目を繰り返すことができることを示します。区切り文字も示されます (それがブランク以外の場合)。

- デフォルトの項目はメインパスの上に表示されます。

- キーワードは、イタリックでない文字で示され、示されているとおりに入力する必要があります。
- 変数は、イタリック体の小文字で示されます。変数は、ユーザー指定の名前や値を表します。
- 句読記号、括弧、算術演算子、または他のそのような記号が表示されている場合は、構文の一部として入力する必要があります。

構文図の例

以下の構文図の例は、**#pragma comment** ディレクティブの構文を示したものです。



注:

1 これが構文図の始まりです。

2 記号 # が最初に示される必要があります。

3 キーワード pragma が # 記号に続いて示される必要があります。

4 プラグマの名前 comment が、キーワード pragma に続いて示される必要があります。

5 左括弧を指定する必要があります。

6 コメント・タイプは、示されているタイプ compiler、date、timestamp、copyright、または user の 1 つとしてのみ入力する必要があります。

7 コメント・タイプ copyright または user とオプショナル文字ストリングとの間にコンマを 1 つ入れる必要があります。

8 コンマの後に文字ストリングが続いている必要があります。文字ストリングは二重引用符で囲む必要があります。

9 右括弧が必要です。

10 これが、構文図の終わりです。

以下の **#pragma comment** ディレクティブの例は、上記の図に従って、構文的に正しいものです。

```
#pragma comment(date)
#pragma comment(user)
#pragma comment(copyright,"This text will appear in the module")
```

本書の例

本書の例は、特に断りのない限り、単純な形式でコーディングされており、ストレージの節約、エラーのチェック、高速パフォーマンスの実現、特定の成果を達成するために使用可能なすべての方法の提示などの試みはなされていません。

インストール情報の例は、*例* または*基本例* としてラベル付けられています。*基本例* は、基本インストールまたはデフォルト・インストール時に実行する手順の説明用です。例はほとんど変更せずに、または全く変更せずに使用できます。

関連情報

以下のセクションでは、XL C/C++ に関連した情報を説明します。

IBM XL C/C++ 情報

XL C/C++ は、以下の形式で製品資料を提供しています。

- README ファイル

  README ファイルには、製品情報に対する変更と訂正も含め、最新の情報が含まれています。 README ファイルは、デフォルトでは XL C/C++ ディレクトリーと、インストール CD のルート・ディレクトリーにあります。

- インストール可能な man ページ

  man ページは製品に準備されているコンパイラー呼び出しとすべてのコマンド行ユーティリティーに対して提供されています。 man ページのインストールおよびアクセスについての指示は、「*IBM XL C/C++ for Linux, V13.1 インストール・ガイド*」に記載されています。

- インフォメーション・センター

検索機能が完備された HTML ベースの資料は、次の Web サイトで参照できます。http://www.ibm.com/support/knowledgecenter/SSXVZZ_13.1.0/com.ibm.compilers.linux.doc/welcome.html

- PDF 文書

  PDF 文書は、デフォルトでは /opt/ibm/xlC/13.1.0/doc/*LANG*/pdf/ ディレクトリーにあります。ここで *LANG* は en_US、zh_CN、 または ja_JP です。 PDF ファイルは、以下の Web サイト http://www.ibm.com/support/docview.wss?uid=swg27036675でも入手できます。

  以下のファイルは、XL C/C++ 製品資料のフル・セットを構成しています。

表 3. XL C/C++ PDF ファイル

文書タイトル	PDF ファイル名	説明
IBM XL C/C++ for Linux, V13.1 インストール・ガイド, SA88-5404-00	install.pdf	XL C/C++ のインストール方法と基本的なコンパイルおよびプログラム実行のための環境の構成方法に関する情報が含まれています。
IBM XL C/C++ for Linux, V13.1 はじめに, SA88-5392-00	getstart.pdf	XL C/C++ 製品の概要と、環境のセットアップと構成、プログラムのコンパイルとリンク、およびコンパイル・エラーのトラブルシューティングに関する情報が含まれています。
IBM XL C/C++ for Linux, V13.1 コンパイラー・リファレンス, SA88-5388-00	compiler.pdf	さまざまなコンパイラー・オプション、プラグマ、マクロ、環境変数、および組み込み関数 (並列処理に使用されるものを含む) についての情報が含まれます。
IBM XL C/C++ for Linux, V13.1 ランゲージ・リファレンス, SA88-5396-00	langref.pdf	移植性および一般的規格への準拠についての言語拡張機能も含め、 IBM によってサポートされる C および C++ プログラミング言語に関する情報が記載されています。
IBM XL C/C++ for Linux, V13.1 最適化およびプログラミング・ガイド, SA88-5402-00	proguide.pdf	アプリケーションの移植、 Fortran コードによる言語間呼び出し、ライブラリー開発、アプリケーションの最適化および並列処理、および XL C/C++ 高性能ライブラリーなどの高度なプログラミング上のトピックに関する情報が記載されています。

  PDF ファイルを読むには、Adobe Reader を使用します。Adobe Reader をお持ちでない場合は、Adobe の Web サイト (http://www.adobe.com) からダウンロードできます (ライセンス条項に従う必要があります)。

IBM Redbooks® 資料、ホワイト・ペーパー、チュートリアル、資料の正誤表、その他の記事など、XL C/C++ に関連する詳細は、次の Web サイトから入手できます。

http://www.ibm.com/support/docview.wss?uid=swg27036675

注: 資料の正誤表は、インフォメーション・センターの英語版にのみ反映されます。

パフォーマンス、生産性、および移植性の向上に関する情報は、 C/C++ café (http://www.ibm.com/software/rational/cafe/community/ccpp) を参照してください。

標準および仕様

XL C/C++ は、以下の標準および仕様をサポートするように設計されています。本情報に含まれているいくつかの機能に関する正確な定義については、これらの標準を参照できます。

- *Information Technology - Programming languages - C, ISO/IEC 9899:1990*、別名 *C89*。
- *Information Technology - Programming languages - C, ISO/IEC 9899:1999*、別名 *C99*。
- *Information Technology - Programming languages - C, ISO/IEC 9899:2011*、別名 *C11*。(部分サポート)
- *Information Technology - Programming languages - C++, ISO/IEC 14882:1998*、別名 *C++98*。
- *Information Technology - Programming languages - C++, ISO/IEC 14882:2003*、別名 *標準 C++*。
- *Information Technology - Programming languages - C++, ISO/IEC 14882:2011*、別名 *C++11*。(部分サポート)
- *Information Technology - Programming languages - Extensions for the programming language C to support new character data types, ISO/IEC DTR 19769.* このドラフトの技術レポートは、C 標準委員会によって承認されており、 http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1040.pdf で入手可能です。
- *Draft Technical Report on C++ Library Extensions, ISO/IEC DTR 19768.* このドラフトの技術レポートは、C 標準委員会に提出されてあり、 http://www.open-std.org/JTC1/SC22/WG21/www/docs/papers/2005/n1836.pdf で入手可能です。
- *AltiVec Technology Programming Interface Manual*, Motorola Inc. ベクトル処理テクノロジーをサポートするための、このベクトル・データ型の仕様はサイト http://www.freescale.com/files/32bit/doc/ref_manual/ALTIVECPIM.pdf で使用可能です。
- *ANSI/IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985.*
- http://www.openmp.org で使用可能な「*OpenMP Application Program Interface Version 4.0 (部分サポート)*」。

その他の IBM 情報

- *ESSL for AIX V5.1/ESSL for Linux on POWER V5.1 Guide and Reference* は、Web ページ Engineering and Scientific Subroutine Library (ESSL) and Parallel ESSL で入手できます。

その他の情報

- *Using the GNU Compiler Collection* は http://gcc.gnu.org/onlinedocs で入手できます。

技術サポート

追加の技術サポートを http://www.ibm.com/support/entry/portal/overview/software/rational/xl_c_for_aixhttp://www.ibm.com/support/entry/portal/overview/software/rational/xl_c~c++_for_linuxhttp://www.ibm.com/support/entry/portal/overview/software/rational/xl_fortran_for_linuxhttp://www.ibm.com/software/awdtools/czos/support/の XL C/C++ のサポート・ページから利用することができます。このページは、選択された大規模な技術情報および他のサポート情報に対する検索機能を備えたポータルを提供します。

必要な情報が見つからない場合は、compinfo@ca.ibm.com に E メールを送信してください。

XL C/C++ に関する最新の情報に関しては、 http://www.ibm.com/software/awdtools/xlc/aix/http://www.ibm.com/software/products/us/en/xlcpp-linux/http://www.ibm.com/software/products/us/en/xlfortran-linuxhttp://www.ibm.com/software/awdtools/czos/にある製品情報サイトをご覧ください。

第 1 章 XL C/C++ の紹介

IBM XL C/C++ for Linux, V13.1 は高機能で高性能なコンパイラーであり、C および Fortran プログラムとの言語間呼び出しなどの、複雑で計算負荷の高い プログラムの開発に使用できます。

このセクションには XL C/C++ のコンパイラーの機能についての概要が記載されています。この章は、コンパイラーの評価を行う方、およびこの製品についてさらに知識を得たい新規ユーザーのために書かれています。

他の IBM コンパイラーとの共通点

IBM XL C/C++ for Linux, V13.1 は、IBM C、C++、Fortran コンパイラーの大規模なファミリーの一部です。 XL C/C++ は、 XL Fortran と一緒になって、XL コンパイラーのファミリーを形成します。

これらのコンパイラーは、種々のプラットフォームおよびプログラミング言語用のコンパイラー機能と最適化テクノロジーを共有する共通のコード・ベースから派生されました。プログラミング環境には、IBM AIX®、IBM Blue Gene®/P、IBM Blue Gene®/Q、IBM i、選択された Linux 配布版、IBM z/OS®、および IBM z/VM® が含まれます。この共通のコード・ベースは、国際的なプログラム言語規格へのコンパイラーの準拠と共に、複数のオペレーティング・システムおよびハードウェア・プラットフォームにわたる整合性のあるコンパイラーのパフォーマンスおよびプログラムの移植の容易さをサポートするのに役立ちます。

オペレーティング・システムのサポート

このセクションでは、 IBM XL C/C++ for Linux, V13.1 がサポートするオペレーティング・システムについて説明します。

IBM XL C/C++ for Linux, V13.1 は以下のオペレーティング・システムをサポートします。

- SUSE Linux Enterprise Server 11 Service Pack 2 (SLES 11 SP2) 以降
- Red Hat Enterprise Linux 6.4 (RHEL 6.4) 以降
- Red Hat Enterprise Linux 7.0 (RHEL 7.0) 以降

要件の完全なリストについては、README ファイルおよび 「*XL C/C++ インストール・ガイド*」の 『XL C/C++ のインストール前の作業』 を参照してください。

IBM XL C/C++ for Linux, V13.1 はビッグ・エンディアン・システムでサポートされています。コンパイラー、そのライブラリー、およびその生成されたオブジェクト・プログラムは、必要なソフトウェアおよびディスク・スペースを備えた POWER5、POWER5+、POWER6®、POWER7®、POWER7+™、および POWER8™ システム上で実行します。

サポートされているさまざまなハードウェア構成を活用するために、コンパイラーにはコンパイルしたアプリケーションを実行するハードウェア・タイプに応じてアプリケーションのパフォーマンスをチューニングするためのオプションがあります。

高度に構成が可能なコンパイラー

多様なコンパイラー呼び出しコマンドおよびオプションを使用して、ユーザーのユニークなコンパイル要件に合うようにコンパイラーを調整できます。

コンパイラー呼び出しコマンド

XL C/C++ には、コンパイラーを呼び出すために使用できるいくつかのコマンドがあります。例えば、**xlC**、**xlc++**、および **xlc** です。各呼び出しコマンドは、特定の言語レベル仕様を満たすためにコンパイル出力を調整するよう、コンパイラーに指示するという点で固有です。コンパイラー呼び出しコマンドは、すべての標準化された C/C++ 言語レベル、および多くの使用頻度の高い拡張機能をサポートするように提供されています。

コンパイラーは、また、大部分の呼び出しコマンドの、対応する「**_r**」バージョンも提供しています。例えば、**xlc_r** および **xlC_r** です。「**_r**」呼び出しは、コンパイラーに、オブジェクト・ファイルをスレッド・セーフのコンポーネントおよびライブラリーにリンクしてバインドするよう指示し、コンパイラーが作成するデータおよびプロシージャーのためのスレッド・セーフのオブジェクト・コードを作成します。

XL C/C++ コンパイラー呼び出しコマンドについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『コンパイラーの呼び出し』 を参照してください。

コンパイラー・オプション

コンパイラーの動作を制御するために、さまざまなコンパイラー・オプションから選択することができます。次の作業を行うために使用できる、便利なオプションがいろいろと用意されています。

- アプリケーションのデバッグ
- アプリケーション・パフォーマンスの最適化およびチューニング
- 他の C または C++ コンパイラーがサポートする非標準の機能および動作との互換性を維持するために、言語レベルおよび言語拡張を選択
- 通常であればソース・コードの変更が必要になるような、他の多くの共通のタスクを実行

環境変数、コンパイラー構成ファイル、コマンド行オプション、およびプログラム・ソースに組み込まれているコンパイラー・ディレクティブ・ステートメントの組み合わせを通じてコンパイラー・オプションを指定できます。

XL C/C++ コンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー・オプション・リファレンス』 を参照してください。

カスタム・コンパイラー構成ファイル

インストール・プロセスは、コンパイラー・オプションのデフォルト設定を定義するスタンザを含む、デフォルトのコンパイラー構成ファイルを作成します。

XL C/C++ のデフォルト設定以外のコンパイラー・オプション設定を頻繁に指定する場合は、Make ファイルを使用してユーザーの設定を定義できます。あるいは、カスタム構成ファイルを作成して自分が頻繁に使用するオプション設定を定義できます。

カスタム・コンパイラー構成ファイルの使用について詳しくは、 57 ページの『カスタム・コンパイラー構成ファイルの使用』を参照してください。

使用状況のトラッキング構成ファイル

この使用状況およびレポート作成ツールを使用して、組織内でのコンパイラーの使用が手持ちのライセンス資格を超過しているかどうかを検出できます。

コンパイラーの使用状況のトラッキングおよびレポート作成機能では、独自の構成ファイルが使用されます。メインのコンパイラー構成ファイルには、そのファイルを指し示す項目が含まれています。コンパイラー製品のさまざまなインストール済み環境で単一の使用状況トラッキング構成ファイルを使用して、使用状況のトラッキングおよびレポート作成機能を集中的に管理できます。

使用状況のトラッキングおよびレポート作成機能について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『コンパイラー使用状況のトラッキングとレポート作成』を参照してください。

言語標準への準拠

IBM XL C/C++ for Linux, V13.1 は、次の C/C++ プログラミング言語仕様をサポートします。

C 言語仕様

- ISO/IEC 9899:2011 (C11 と呼ばれる) の部分的サポート
- ISO/IEC 9899:1999 (C99 と呼ばれる)
- ISO/IEC 9899:1990 (C89 と呼ばれる)

C++ 言語仕様

- ISO/IEC 14882:2011 (C++11 と呼ばれる) の部分的サポート
- ISO/IEC 14882:2003 (標準 C++ と呼ばれる)
- ISO/IEC 14882:1998、この言語の最初の正式の仕様 (C++98 と呼ばれる)

標準の言語レベルに加え、 XL C/C++ は次の言語拡張機能をサポートします。

- OpenMP アプリケーション・プログラム・インターフェース V4.0 の部分的サポート
- OpenMP Application Program Interface V3.1
- ベクトル・プログラミングをサポートする言語拡張
- GNU C および C++ 言語拡張機能のサブセット

C/C++ 言語仕様および拡張について詳しくは、 「*XL C/C++ ランゲージ・リファレンス*」の『言語レベルおよび言語拡張』および「*XL C/C++ ランゲージ・リファレンス*」の『言語標準』を参照してください。

GNU との互換性

XL C/C++ は、 **gcc** コンパイラーおよび **g++** コンパイラーを使用して開発されたアプリケーションの移植を容易にするために、 GNU コンパイラー・コマンド・オプションのサブセットをサポートします。

このサポートは、**gxlc** または **gxlc++** の呼び出しコマンドを一部の GNU コンパイラー・オプションと組み合わせて使用した場合に利用できます。可能な場合は、コンパイラーを呼び出す前に、コンパイラーは GNU オプションを、それぞれに対応する XL C/C++ コンパイラー・オプションにマップします。

呼び出しコマンドは、プレーン・テキストの構成ファイルを使用して GNU-to-XL C/C++ オプション・マッピングおよびデフォルトを制御します。この構成ファイルをカスタマイズしてユーザーのユニークなコンパイル要件を満たすことができます。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『gxlc および gxlc++ による GNU C/C++ コンパイラー・オプションの再使用』を参照してください。

XL C/C++ は、GNU C および GNU C++ ヘッダー・ファイルを、GNU C および C++ ランタイム・ライブラリーと一緒に使用して、GNU Compiler Collection (GCC) が生成したコードとバイナリー互換のコードを生成します。アプリケーションの一部は、XL C/C++ を使用してビルドすることができ、GCC を使用してビルドされた部分と結合して、GCC のみを使用してビルドされたかのように動作するアプリケーションを作成することができます。

GCC でコンパイルされたコードとのバイナリー互換性を実現させるため、XL C/C++ でコンパイルされたプログラムには、同じシステム上にある GNU コンパイラーによって使用されるものと同じヘッダーが組み込まれます。正しいバージョンのヘッダーおよびランタイム・ライブラリーがシステム上に存在することを確実にするために、 XL C/C++ をインストールするには、事前に前提条件の GCC コンパイラーがインストールされている必要があります。

この関係に関する、いくつかの注意すべき追加項目を以下に挙げます。

- IBM 組み込み関数は、GNU C 組み込み関数と共存する。
- C および C++ プログラムのコンパイルは、GNU C および GNU C++ ヘッダー・ファイルを使用する。
- コンパイルは、アセンブラー入力ファイルに対して GNU アセンブラーを使用する。
- コンパイルされた C コードは、GNU C ランタイム・ライブラリーにリンクされる。
- コンパイルされた C++ コードは、GNU C、および GNU C++ ランタイム・ライブラリーにリンクされる。
- デバッグは GNU デバッガー **gdb** を使用する。

ソース・コード・マイグレーションおよび規格合致検査

XL C/C++ には、特定の言語レベルにユーザーのアプリケーション・コードをコンパイルするようにコンパイラーに指示するコンパイラー呼び出しコマンドがあります。

ユーザーは、**-qlanglvl** コンパイラー・オプションを使用して、言語レベルを指定することもできます。コンパイラーは、プログラム・ソース内の言語または言語拡張エレメントが、指定された言語レベルに準拠していない場合、診断メッセージを出します。

詳しくは、 「*XL C/C++ コンパイラー・リファレンス*」の『-qlanglvl』を参照してください。

ライブラリー

XL C/C++ には、さまざまなライブラリーが入っているランタイム環境が組み込まれています。

Mathematical Acceleration Subsystem ライブラリー

Mathematical Acceleration Subsystem (MASS) ライブラリーは、特に、サポートされたプロセッサー・アーキテクチャーでの最適なパフォーマンス用に調整された、スカラーおよびベクトルの数学組み込み関数から成り立っています。 さまざまなプロセッサーでの高性能コンピューティングをサポートする MASS ライブラリーを選択するか、または特定のプロセッサー・ファミリーをサポートするようチューニングされたライブラリーを選択できます。

MASS ライブラリー関数は、32 ビットおよび 64 ビットの両方のコンパイル・モードをサポートし、そのパフォーマンスはデフォルトの `libm` 数学ライブラリー・ルーチンよりも向上しています。これらのライブラリーはスレッド・セーフであり、ユーザーが自身のアプリケーション用に特定のレベルの最適化を要求したとき自動的に呼び出されます。ユーザーは、また、最適化オプションが有効であるかないかに関係なく、MASS ライブラリー関数への明示的呼び出しを行うことができます。

詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『Mathematical Acceleration Subsystem の使用』を参照してください。

Basic Linear Algebra Subprograms

高性能代数関数の Basic Linear Algebra Subprograms (BLAS) セットは、libxlopt ライブラリーに入れて配送されます。 これらの関数を使用して、以下のことを行うことができます。

- 汎用行列またはその転置用の行列ベクトルの積を計算します。
- 汎用行列またはそれらの転置用の複合行列の乗算および追加を実行します。

BLAS 関数の使用に関する詳細については、 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『Basic Linear Algebra Subprograms の使用』を参照してください。

その他のライブラリー

次のライブラリーも XL C/C++ に標準装備されています。

- SMP Runtime Library は、明示的並列処理および自動化された並列処理の両方をサポートします。「*XL C/C++ 最適化およびプログラミング・ガイド*」の『SMP Runtime Library』を参照してください。
- XL C++ Runtime Library には、コンパイラーが必要とするサポート・ルーチンが入っています。

Boost ライブラリーのサポート

IBM XL C/C++ for Linux, V13.1 は Boost V1.55.0 ライブラリーを部分的にサポートしています。また、Boost V1.55.0 ライブラリーを変更して、 XL C/C++ アプリケーションでビルドまた使用できるようにするための、パッチ・ファイルが使用可能です。このパッチまたは変更ファイルは、Boost ライブラリーの機能を拡張したり追加したりするものではありません。

Boost ライブラリーをビルドするためのパッチ・ファイルを入手するには 「Boost Library Regression Test Summaries」 を参照し、使用するコンパイラー・リリースおよびプラットフォームの「`download required Boost modification file`」 を選択してください。

最新の Boost ライブラリーは、http://www.boost.org/ でダウンロードできます。

ライブラリーのサポートについて詳しくは、XL C/C++ コンパイラーのサポート・ページ (http://www.ibm.com/support/entry/portal/overview/software/rational/xl_c~c++_for_linux) で検索してください。

ツール、ユーティリティー、およびコマンド

このトピックでは、XL C/C++ に組み込まれている主なツール、ユーティリティー、およびコマンドについて説明します。一部のコンパイラー・ツール、ユーティリティー、およびコマンドについては説明していません。

ツール

使用状況レポート作成ツール

使用状況レポート作成ツールは、組織によるコンパイラーの使用状況を記述したレポートを生成します。これらの機能は、組織内でのコンパイラーの使用がコンパイラー・ライセンス資格に一致するかどうかを判断するのに役立ちます。 **urt** コマンドには、レポートのカスタマイズに使用できるオプションがあります。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー使用状況トラッキングおよびレポート作成』を参照してください。

ユーティリティー

gxlc および gxlc++ ユーティリティー

gxlc および **gxlc++** ユーティリティーは、GNU C および GNU C++ の呼び出しコマンドを対応する **xlc** および **xlc++** のコマンドに変換してから XL C/C++ コンパイラーを呼び出します。これらのユーティリティーの目的

は、GNU コンパイラーで作成された既存のアプリケーション用に使用される makefile への変更の数を最小化し、XL C/C++ コンパイラーへの移行を容易にすることにあります。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『gxlc および gxlc++ での GNU C/C++ コンパイラー・オプションの再使用』を参照してください。

new_install

new_install ユーティリティーは、コンパイラーがインストールされた後に、ご使用のシステムで使用するために IBM XL C/C++ for Linux, V13.1 を構成します。13.1 以降では、XL C/C++ と IBM Advance Toolchain との併用を容易にするために **new_install** ユーティリティーを使用できます。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『Advance Toolchain との IBM XL C/C++ for Linux, V13.1 の併用』を参照してください。

xlc_configure

xlc_configure ユーティリティーは、ユーザー独自のコンパイラー・オプションのデフォルト設定値のカスタム・セットを入れる、追加コンパイラー構成ファイルを作成します。詳しくは、「*XL C/C++ インストール・ガイド*」の 『xlc_configure ユーティリティーを直接実行する (上級者向け)』を参照してください。

コマンド

genhtml コマンド

genhtml コマンドは、**-qlistfmt** オプションによって生成された既存の XML 診断レポートを変換します。**-qlistfmt** オプションを使用することによって、XML と HTML のいずれの診断レポートを生成するかを選択できます。レポートは、最適化の機会を検出するのに役立ちます。このコマンドの使用方法について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『**genhtml** コマンド』を参照してください。

Profile-Directed Feedback (PDF) 関連のコマンド

cleanpdf コマンド

cleanpdf コマンドは、 Profile-Directed Feedback データが書き込まれるディレクトリーからすべての PDF ファイルまたは指定された PDF ファイルを削除します。

mergepdf コマンド

mergepdf コマンドは、複数の PDF レコードを単一のレコードに結合するときに、それらのレコードの重要性を評価する機能を提供します。PDF レコードは、同じ実行可能モジュールから得られたものである必要があります。

resetpdf コマンド

resetpdf コマンドの現在の動作は、**cleanpdf** コマンドと同じであり、他のプラットフォーム上での前のリリースとの互換性のために保持されています。

showpdf コマンド

showpdf コマンドは、PDF 実行 (**-qpdf1** オプションを使用したコ

ンパイル) で実行されたすべてのプロシージャーについて、以下の種類のプロファイル情報を表示します。

- ブロック・カウンター・プロファイル
- 呼び出しカウンター・プロファイル
- 値プロファイル
- キャッシュ・ミス・プロファイル (**-qpdf1** フェーズで **-qpdf1=level=2** オプションを指定した場合)。

最初の 2 種類のプロファイル情報は、テキストまたは XML 形式のいずれかで表示できます。しかし、値プロファイルおよびキャッシュ・ミス・プロファイル情報は XML 形式でしか表示できません。

詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qpdf1、-qpdf2』を参照してください。

プログラムの最適化

XL C/C++ は、プログラムの最適化およびパフォーマンスを制御するのに役立つ可能性のあるいくつかのコンパイラー・オプションを提供します。

これらのオプションを使用して、以下の作業を実行できます。

- さまざまなレベルのコンパイラーの最適化を選択する。
- ループ、浮動小数点、その他のタイプの操作に関する最適化を制御する。
- プログラムが実行される場所に応じて、プログラムを、特定のクラスのマシンまたは非常に特定度の高いマシン構成に合わせて最適化する。

変換の最適化によって、アプリケーションの全体的な実行パフォーマンスを向上させることができます。 XL C/C++ は、サポートされるさまざまなハードウェアに合わせた変換の最適化のポートフォリオを提供します。このような変換では、以下の利点があります。

- クリティカルな操作に対して実行する命令の数の削減。
- Power アーキテクチャー を最大限活用するための、生成済みオブジェクト・コードの再構築。
- メモリー・サブシステムの使用法の改善。
- 大量の共用メモリー並列処理を扱うアーキテクチャーの能力の活用。

詳しくは、以下の関連トピックを参照してください。

- 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『アプリケーションの最適化』
- 「*XL C/C++ コンパイラー・リファレンス*」の『最適化およびチューニング・オプション』
- 「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー組み込み関数』

64 ビット・オブジェクトの機能

XL C/C++ コンパイラーの 64 ビット・オブジェクトの機能は、より大きいストレージ要件およびより強力な処理能力に対する増大する要求に対処するものです。

Linux オペレーティング・システムは、64 ビットのアドレス・スペースの使用を通じて 64 ビットのプロセッサーを活用するプログラムを開発し、実行できるようにする環境を提供します。

64 ビット・アドレス・スペース内に収めることのできるより大きな実行可能ファイルをサポートするために、別個の 64 ビット・オブジェクト形式が使用されます。リンカーはこれらのオブジェクトをバインドして、64 ビット実行可能ファイルを作成します。一緒にバインドされるオブジェクトは、すべて同じオブジェクト形式になっている必要があります。 以下のシナリオは許されず、ロード、実行、あるいはその両方に失敗します。

- 32 ビット・ライブラリーまたは共用ライブラリーからの、シンボルへの参照をもっている 64 ビット・オブジェクトまたは実行可能モジュール
- 64 ビット・ライブラリーまたは共用ライブラリーからの、シンボルへの参照をもっている 32 ビット・オブジェクトまたは実行可能モジュール
- 32 ビット・モジュールを明示してロードしようとする 64 ビット実行可能モジュール
- 64 ビット・モジュールを明示してロードしようとする 32 ビット実行可能モジュール

XL C/C++ は、主に **-q64** コンパイラー・オプションおよび **-qarch** コンパイラー・オプションを使用することによって 64 ビット・モードをサポートします。この組み合わせは、目標のアーキテクチャー用のビット・モードと命令セットを決めます。

詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の 『32 ビットおよび 64 ビット・モードの使用』を参照してください。

共用メモリーの並列処理

XL C/C++ は、マルチプロセッサー・システム体系についてのアプリケーション開発をサポートします。

XL C/C++ での並列化アプリケーションの開発には、以下に挙げるいずれの方法でも使用することができます。

- ディレクティブ・ベースの共用メモリー並列処理
- コンパイラーへ、共用メモリー並列処理を自動的に生成するよう指示する
- メッセージ引き渡しベースの共用または分散メモリー並列処理 (MPI)

詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『プログラムの並列処理』 を参照してください。

OpenMP ディレクティブ

OpenMP ディレクティブは、XL C/C++ および他の多くの IBM および IBM 以外の C、C++、および Fortran コンパイラーによってサポートされる API ベースのコマンドのセットです。

ユーザーは、OpenMP ディレクティブを使用して、特定のループを並列化する方法をコンパイラーに指示することができます。ソースにディレクティブがあると、コンパイラーが並列コードに対して並列分析を実行する必要がなくなります。
OpenMP ディレクティブは、並列処理に必要なインフラストラクチャーを提供する Pthread ライブラリーの存在を必要とします。

OpenMP ディレクティブは、アプリケーションを並列化するための重要な 3 つの問題に対処します。

1. 節 (clause) およびディレクティブは、変数のスコーピングに使用できます。変数を共用すべきではない場合が頻繁に起こります。いいかえれば、プロセッサーはそれぞれ、それ自身の変数のコピーを持っている必要があります。
2. 作業共用ディレクティブは、コードの並列領域に入っている作業を複数のプロセッサー間で分散させる方法を指定します。
3. ディレクティブは、複数のプロセッサー間での同期を制御するのに使用できます。

IBM XL C/C++ for Linux, V13.1 から、XL C/C++ は、 OpenMP API バージョン 3.1 および OpenMP API バージョン 4.0 仕様から選ばれた機能をサポートします。詳しくは、 16 ページの『OpenMP 4.0』を参照してください。

プログラムのパフォーマンスの最適化について詳しくは、以下を参照してください。

- 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『アプリケーションの最適化』
- 『The OpenMP API specification for parallel programming』

診断リスト

コンパイラー出力リストおよび XML または HTML レポートから、アプリケーションをより効率的に開発したりデバッグするのに役立つ重要な情報が得られます。

リストされる情報は、組み込むことも、あるいは省略することもできる、任意のセクションで構成されています。適用可能なコンパイラー・オプションおよびリスト作成について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー・メッセージおよびリスト表示』を参照してください。

また、コンパイラーが行った、または行わなかった最適化の一部に関する診断情報を、コンパイラーから XML または HTML の形式で取得できます。この情報を使用すると、アプリケーション (特に高性能アプリケーション) をチューニングするときのプログラミングの労力を抑えることができます。このレポートは XML スキーマによって定義され、ユーザーが結果の読み取りと分析のために作成できるツールで簡単に取り込むことができます。このレポート、およびレポートの使用法に関し

て詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『レポートを使用した最適化の機会の診断』を参照してください。

シンボリック・デバッガー・サポート

さまざまなレベルの **-g** コンパイラー・オプションを使用することで、コンパイルしたオブジェクトにデバッグ情報を組み込むように、XL C/C++ に指示できます。

詳しくは、**-g**を参照してください。

デバッグ情報は **gdb** またはその他の任意のシンボリック・デバッガーによって調べることができ、プログラムのデバッグに役立ちます。

第 2 章 IBM XL C/C++ for Linux, V13.1 の新機能

このセクションでは、 IBM XL C/C++ for Linux, V13.1 に追加された機能と機能拡張について説明します。

POWER8 プロセッサーのサポート

XL C/C++ for Linux, V13.1 は、POWER8 プロセッサーをサポートします。

POWER8 プロセッサーのサポートで導入された新機能および機能拡張は、以下のカテゴリーに分類されます。

- POWER8 プロセッサー用の MASS ライブラリー
- POWER8 プロセッサー用のコンパイラー・オプション
- POWER8 プロセッサー用の組み込み関数

POWER8 プロセッサー用の Mathematical Acceleration Subsystem (MASS) ライブラリー

ベクトル・ライブラリー

ベクトル MASS ライブラリー **libmassvp8.a** には、POWER8 アーキテクチャー用に調整されたベクトル関数が入っています。これらの関数は、32 ビットと 64 ビットのどちらのモードでも使用できます。

ベクトル・ライブラリーについて詳しくは、 『ベクトル・ライブラリーの使用』(「*XL C/C++ 最適化およびプログラミング・ガイド*」内) を参照してください。

SIMD ライブラリー

MASS SIMD ライブラリー **libmass_simdp8.a** には、頻繁に使用される組み込み数学関数の高速セットが含まれています。これらは、対応する標準システム・ライブラリーの関数を上回るパフォーマンスを発揮します。

SIMD ライブラリーについて詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『SIMD ライブラリーの使用』を参照してください。

POWER8 プロセッサー用のコンパイラー・オプション

-qarch コンパイラー・オプションはコードの生成対象であるプロセッサー・アーキテクチャーを指定します。**-qtune** コンパイラー・オプションは、特定のハードウェア・アーキテクチャーで最も効率よく実行できるように、命令選択、スケジューリング、およびその他のアーキテクチャー依存のパフォーマンス拡張機能をチューニングします。

新しい **-qarch=pwr8** サブオプションは、 POWER8 ハードウェア・プラットフォーム上で実行される命令が含まれるオブジェクト・コードを生成します。新しい **-qtune=pwr8** サブオプションを使用すると、POWER8 ハードウェア・プラットフォームに合わせて、最適化のための調整が行われます。

詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qarch』および「*XL C/C++ コンパイラー・リファレンス*」の『-qtune』を参照してください。

POWER8 プロセッサー用の組み込み関数

以下の POWER8 プロセッサー機能をサポートするために、新しいハードウェア組み込み機能が追加されました。

- POWER8 ベクトル処理用の組み込み機能
- POWER8 2 進化 10 進関数
- POWER8 暗号化関数
- POWER8 4 倍長ワード算術関数
- POWER8 load-and-reserve 命令および store conditional 命令
- POWER8 キャッシュおよびデータ・プリフェッチ制御関数
- POWER8 トランザクション・メモリー関数
- POWER8 プリフェッチ関数

XL C/C++ に用意されている組み込み関数について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー組み込み関数』を参照してください。

Advance Toolchain 7.0 のサポート

IBM XL C/C++ for Linux, V13.1 は IBM Advance Toolchain 7.0 を完全にサポートします。これはオープン・ソースの、開発ツールおよびランタイム・ライブラリーのセットです。 IBM Advance Toolchain を使用すると、 Linux 上で最新の POWER® ハードウェア機能、特にチューニングされたライブラリーを利用できます。

詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『Advance Toolchain との IBM XL C/C++ for Linux, V13.1 の併用』を参照してください。

C++11 機能

既存の C++11 機能に加えて、このリリースの XL C/C++ では新しい C++11 機能がサポートされています。

注: IBM は、承認される以前は C++0x として知られていた C++11 の、選ばれた機能をサポートします。IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C++11 標準ライブラリーのサポートを含め、 C++11 のすべての機能を IBM が実装し終えるまで、リリースごとに実装が変更される可能性があります。IBM では、ソース、バイナリー、またはリストおよび他のコンパイラー・インターフェースにおいて、新しい C++11 機能の IBM による以前のリリース実装との互換性を維持する試みは行いません。

XL C/C++ V13.1 には、以下の機能が導入されています。

- デフォルト関数および削除済み関数
- `nullptr` キーワード

XL C/C++ V13.1 では、一般化された定数式の機能が拡張されました。

-qlanglvl=extended0x オプションを使用すると、ほとんどの C++ の機能と、現在サポートされているすべての C++11 の機能を使用可能にすることができます。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qlanglvl』を参照してください。

デフォルト関数および削除済み関数

この機能では、2 つの新しい形式の関数宣言を導入して、明示的にデフォルトに設定された関数および削除済み関数を定義します。コンパイラーは、明示的にデフォルトに設定された関数について、デフォルトの実装を生成し、これは手動でプログラムした実装より効率的です。また、コンパイラーは、不要な関数を呼び出さないよう、削除済み関数を使用不可にします。

-qlanglvl=defaultanddelete オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『明示的にデフォルトに設定された関数 (C++11)』および『削除済み関数 (C++11)』を参照してください。

一般化された定数式

一般化された定数式の機能により、定数式内で使用できる一連の式が拡張されます。XL C/C++ V12.1 でのこの機能の実装は、 C++11 標準で定義されている機能を部分的に実装したものでした。このリリースでは、ユーザー定義の `constexpr` オブジェクトおよび `constexpr` ポインター、または `constexpr` 関数およびオブジェクトへの参照をサポートするように機能拡張されています。

-qlanglvl=constexpr オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『一般化された定数式 (C++11)』を参照してください。

nullptr キーワード

この機能は、 `nullptr` を NULL ポインター定数として導入します。 `nullptr` 定数は、多重定義関数の整数 0 とは区別できます。0 定数および NULL 定数は、多重定義関数の整数型として扱われる一方で、`nullptr` はポインター型、pointer-to-member 型、およびブール型のみに、暗黙的に変換できます。

-qlanglvl=nullptr オプションを使用してこの機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『langref.pdf#nullpt』を参照してください。

「*XL C/C++ コンパイラー・リファレンス*」**内の関連情報**

-qlanglvl

C11 機能

このリリースの XL C/C++ では、C11 の既存の機能に加えて C11 の新機能がサポートされています。

注: IBM は、C11 の選択された機能 (C1X と呼ばれる) をその承認の前にサポートします。 IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C11 標準ライブラリーのサポートを含め、すべての C11 機能を IBM が実装し終えるまで、リリースごとに実装が変更される可能性があります。IBM では、IBM による新規 C11 機能の実装に関し、ソース、バイナリー、リスト作成などのコンパイラー・インターフェースにおいて、以前のリリースとの互換性を維持するための試みは特に行いません。

IBM XL C/C++ for Linux, V13.1 には、以下の機能が導入されています。

- `typedef` 再宣言
- 汎用選択

typedef 再宣言

`typedef` 再宣言を使用し、同じスコープ内で以前の `typedef` 名である名前を再定義して同じ型を参照できます。 IBM XL C コンパイラーは、可変的な型を含むすべての型をサポートします。 IBM 詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『typedef 定義』を参照してください。

汎用選択

汎用選択により、コンパイル時に指定される型名に応じて式を選択する仕組みが提供されます。一般的には、型汎用マクロを定義するために使用します。詳しくは、『汎用選択 (Generic selection) (C11)』を参照してください。

OpenMP 4.0

IBM XL C/C++ for Linux, V13.1 では、 OpenMP アプリケーション・プログラム・インターフェース・バージョン 4.0 仕様の一部をサポートします。XL C/C++ 実装は、OpenMP アプリケーション・プログラム・インターフェース・バージョン 4.0 の IBM の解釈に基づいています。

このバージョンの XL C/C++ は、以下の OpenMP 4.0 機能をサポートします。

- `update` 節および `capture` 節の機能拡張
- OMP_DISPLAY_ENV 環境変数

update 節および capture 節の機能拡張

より多くの式形式をサポートするため、`atomic` 構文の `update` 節および `capture` 節が拡張されました。

OMP_DISPLAY_ENV 環境変数

OMP_DISPLAY_ENV 環境変数を使用して、環境変数に関連付けられた内部制御変数 (ICV) の値、およびランタイム・ライブラリーに関するビルド固有の情報を表示できます。

関連情報

- 「*XL C/C++ コンパイラー・リファレンス*」の『OpenMP 環境変数』
- 「*XL C/C++ コンパイラー・リファレンス*」の『並列処理のためのプラグマ・ディレクティブ』
- 『The OpenMP API specification for parallel programming』

組み込み関数

次の主要なカテゴリーの組み込み関数はこのリリースの新機能です。

ベクトル処理用の POWER8 組み込み関数

次のベクトル組み込み関数が追加されています。

- ベクトルのバイト単位ビット集約ダブルワード関数
  - `vec_gbb`
- ベクトル先行ゼロ・カウント関数
  - `vec_cntlz`
- ベクトル・ポピュレーション・カウント関数
  - `vec_popcnt`
- 拡張ベクトル論理演算関数
  - `vec_eqv`
  - `vec_nand`
  - `vec_orc`
- 128 ビット整数の加算および減算関数
  - `vec_add_u128`
  - `vec_sub_u128`
  - `vec_adde_u128`
  - `vec_sube_u128`
  - `vec_addc_u128`
  - `vec_subc_u128`
  - `vec_addec_u128`
  - `vec_subec_u128`
  - `vec_bperm`

以下の組み込み関数が拡張され、ダブルワード・タイプがサポートされるようになりました。

- ベクトル PACK 関数
  - `vec_pack`

  - vec_packs
  - vec_packsu
- ベクトル UNPACK 関数
  - vec_unpackh
  - vec_unpackl
- ベクトル加算および減算関数
  - vec_add
  - vec_sub
- ベクトル MAX および MIN 関数
  - vec_max
  - vec_min
- ベクトル・シフトおよび回転関数
  - vec_rl
  - vec_sl
  - vec_sr
  - vec_sra
- ベクトル比較関数
  - vec_cmpeq
  - vec_cmpgt
  - vec_cmpge
  - vec_cmplt
  - vec_cmple

POWER8 2 進化 10 進組み込み関数

次の組み込み関数が追加され、2 進化 10 進 (BCD) 算術および比較がサポートされました。

- BCD 加算および減算関数
  - __bcdadd
  - __bcdsub
- オーバーフローの BCD テスト加算および減算関数
  - __bcdadd_ofl
  - __bcdsub_ofl
  - __bcd_invalid
- BCD 比較関数
  - __bcdcmpeq
  - __bcdcmpgt
  - __bcdcmpge
  - __bcdcmplt
  - __bcdcmple
- BCD ロードおよび保管関数

  - __vec_ldrmb
  - __vec_strmb

POWER8 暗号化組み込み関数

暗号操作を実行するために、次の組み込み関数が提供されています。

- Advanced Encryption Standard (AES) 関数
  - __vcipher
  - __vcipherlast
  - __vncipher
  - __vncipherlast
  - __vsbox
- Secure Hash Algorithm (SHA) 関数
  - __vshasigmad
  - __vshasigmaw
- その他の関数
  - __vpmsumb
  - __vpmsumh
  - __vpmsumw
  - __vpmsumd
  - __vpermxor

POWER8 非ベクトル組み込み関数

キャッシュの効率を向上するための、次の組み込み関数が追加されました。

- __dcbtna
- __icbt

ロードおよび保管組み込み関数では、より多くの型をサポートするよう次の関数が拡張されました。

- __lqarx
- __lharx
- __lbarx
- __stqcx
- __sthcx
- __stbcx

POWER8 トランザクション・メモリー組み込み関数

トランザクション・メモリーは並列プログラミングのモデルです。このモデルでは、アトミックとして処理される命令またはステートメントのブロックを指定できます。

次の組み込み関数を使用してトランザクションの始まりまたは終わりにマークを付けたり、失敗の理由を診断したりできます。

- トランザクション開始および終了関数
  - __TM_begin
  - __TM_end
  - __TM_simple_begin
- トランザクション打ち切り関数
  - __TM_abort
  - __TM_named_abort
- トランザクション照会関数
  - __TM_failure_address
  - __TM_failure_code
  - __TM_is_conflict
  - __TM_is_failure_persistent
  - __TM_is_footprint_exceeded
  - __TM_is_illegal
  - __TM_is_named_user_abort
  - __TM_is_nested_too_deep
  - __TM_is_user_abort
  - __TM_nesting_depth

POWER8 プリフェッチ組み込み関数

次の組み込み関数は、データ・ストリーム制御レジスター (DSCR) の問題プログラム状態制御を、直観的で、移植可能、かつ最適化しやすい形で表示します。

- 一時的属性の有効化関数
  - __hardware_transient_enable
  - __load_transient_enable
  - __software_transient_enable
  - __store_transient_enable
- ユニット・カウントの有効化および設定関数
  - __hardware_unit_count_enable
  - __software_unit_count_enable
  - __set_prefetch_unit_count
- プリフェッチの深さ関数
  - __default_prefetch_depth
  - __depth_attainment_urgency
- ロード・ストリーム有効化および無効化関数
  - __load_stream_disable
  - __stride_n_stream_enable
- DSCR 関数
  - __prefetch_get_dscr_register
  - __prefetch_set_dscr_register

注: POWER8 組み込み関数は、**-qarch** が POWER8 プロセッサーをターゲットにするよう設定されているときにのみ有効です。

XL C/C++ に用意されている組み込み関数について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー組み込み関数』を参照してください。

コンパイラー・オプションおよびプラグマ・ディレクティブ

このセクションでは、新規または変更されたコンパイラー・オプションおよびプラグマ・ディレクティブについて説明します。

コマンド行でコンパイラー・オプションを指定することができます。また、アプリケーション・ソース・ファイルに埋め込まれたプラグマ・ディレクティブによりコンパイラーの動作を変更することもできます。XL C/C++ コンパイラー・オプションの詳細説明および使用法に関する情報については、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。

-qarch オプションのデフォルトは **pwr5** に更新されました。古いハードウェア・ファミリーを示すサブオプションは、自動的により新しいアーキテクチャーにアップグレードされます。

以下のサブオプションが追加または更新されています。

-qarch=pwr7
このサブオプションでは、 POWER7、POWER7+、または POWER8 ハードウェア・プラットフォームで実行する命令を含んだオブジェクト・コードを生成します。

-qarch=pwr8
このサブオプションでは、POWER8 ハードウェア・プラットフォームで実行する命令を含んだオブジェクト・コードを作成します。

-qcheck
以下のサブオプションが追加または更新されています。

-qcheck=stackclobber
このサブオプションでは、ユーザーのプログラム内の特定タイプのスタック破損を検出します。

-qcheck=unset
このサブオプションでは、実行時に設定される前に使用されている自動変数を検査します。

-qdbgfmt
次のサブオプションが追加されています。

-qdbgfmt=dwarf
このサブオプションではデバッグ情報を DWARF 3 形式で作成します。

-qdbgfmt=dwarf4
このサブオプションではデバッグ情報を DWARF 4 形式で作成します。

-qhelp このオプションでは、コンパイラーの man ページを表示します。

-qinfo

コンパイラーは、以下のファイルについて情報メッセージを出しません。

- コンパイラーおよびシステム・ヘッダー・ファイルの標準検索パス内のファイル。
- コンパイラーおよびシステム・ヘッダー・ファイルの標準検索パス内のファイルによって最終的に組み込まれるファイル。

以下のサブオプションが追加または更新されています。

-qinfo=mt
このサブオプションでは、同期が必要な可能性のある場所について通知します。

-qinfo=unset
このサブオプションでは、設定される前に使用されている自動変数を検出して、コンパイル時に通知メッセージのフラグを立てます。

-qlanglvl

以下のサブオプションが追加または更新されています。

C++11 **-qlanglvl=defaultanddelete**
このサブオプションでは、デフォルトに設定された関数および削除済み関数の機能を使用可能にします。ユーザーは、このサブオプションを使用して、より高い効率を実現する目的でコンパイラーにより実装が生成されるデフォルト関数を明示的に定義できます。また、この機能を使用して、不要な関数を呼び出さない目的でコンパイラーにより使用不可になっている削除済み関数を定義できます。
C++11

C++11 **-qlanglvl=nullptr**
このサブオプションにより `nullptr` 機能が有効になります。この機能を使用すると、`nullptr` 定数で NULL ポインターを初期化できます。 NULL ポインターは、ポインター型、pointer-to-member 型、またはブール型に変換できます。`nullptr` 定数は、多重定義関数の整数 0 とは区別できます。 C++11

-qpdf1=unique
このサブオプションでは、実行時にプロセスごとに固有の PDF ファイルを作成します。

-qprefetch=dscr
このサブオプションは、アプリケーションの実行時のパフォーマンスを向上させるのに役立ちます。システム・アーキテクチャーに応じて、`dscr` の値を指定できます。

-qsimd=auto
このサブオプションは、非推奨の **-qhot=simd** オプションによって実行されていた自動 SIMD 化を制御します。

-qstaticlink=xllibs
このサブオプションは XL コンパイラー・ライブラリーで静的にリンクします。

-qtune オプションのデフォルトは更新されました。

以下のサブオプションが追加または更新されています。

-qtune=pwr7
このサブオプションでは、POWER7 または POWER7+ ハードウェア・プラットフォームに合わせて最適化が調整されることを指定します。

-qtune=pwr8
このサブオプションでは、POWER8 ハードウェア・プラットフォームに合わせて最適化が調整されることを指定します。

SMT サブオプション
新しい **-qtune** 同時マルチスレッド化 (SMT) サブオプションでは、ターゲット SMT を指定して、最適化によりそのモードで最高のパフォーマンスを得られるよう指示することができます。

-qunroll=*n*
このサブオプションでは、ループを *n* 回アンロールするようコンパイラーに示唆します。ループの反復が *n* 回より少ない場合は、完全にアンロールされます。

-qvisibility
このオプションは、エンティティーの可視属性を指定します。エンティティーの可視属性は、あるモジュールに定義されたエンティティーを他のモジュールで参照または使用できるかどうか、あるいはその方法について記述します。visibility 属性は、外部リンケージを使用するエンティティーにのみ影響し、その他のエンティティーの可視性を追加することはできません。

新規または変更されたプラグマ・ディレクティブ

#pragma GCC visibility push, #pragma GCC visibility pop
プラグマ・ディレクティブのこのペアは、**-qvisibility** オプションのプラグマ版です。プラグマ・ディレクティブは、外部リンケージ・シンボルの可視属性を指定するため使用されます。

パフォーマンスおよび最適化

追加の機能および機能拡張を、パフォーマンスの調整とアプリケーションの最適化に役立てることができます。

IBM

エンティティーの可視属性

エンティティーの可視属性は、あるモジュールに定義されたエンティティーを他のモジュールで参照または使用できるかどうか、あるいはその方法について記述します。エンティティーの可視属性を使用すると、以下の利点が得られます。

- 共用ライブラリーのサイズ縮小
- シンボル競合の可能性低減
- コンパイル・フェーズおよびリンク・フェーズでの最適化向上が可能
- 動的リンクの効率向上

詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『 visibility 属性の使用 (IBM 拡張)』を参照してください。

IBM

パフォーマンス調整およびプログラム最適化について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『アプリケーションの最適化』および『パフォーマンスを向上させるためのアプリケーションのコーディング』を参照してください。

第 3 章 以前のバージョンからのマイグレーション

コンパイラーの最新バージョンにマイグレーションすることにより、コンパイラーの新機能 (コンパイルするアプリケーションのパフォーマンスを向上させる機能など) を活用できます。

より新しいバージョンのコンパイラーには、新機能および拡張機能が組み込まれています。これらの機能には、以下のメリットがあります。

- 追加のパフォーマンス強化による、コンパイルするアプリケーションの最適化の向上
- 複数のオペレーティング・システムおよびハードウェア・プラットフォーム間でのコードの移植性を促進する、新しい言語標準のサポート
- 新しいハードウェアおよびオペレーティング・システムの最新機能の利用

コンパイラーの最新バージョンにマイグレーションすることにより、向上したパフォーマンス最適化、新しい言語規格のサポート、新しいハードウェアおよびソフトウェア環境の利用などの、新機能の利点を享受することができます。

現行リリースの新機能について詳しくは、 13 ページの『第 2 章 IBM XL C/C++ for Linux, V13.1 の新機能』を参照してください。ホワイト・ペーパー『*XL C/C++ コンパイラーのアップグレード (Upgrading XL C/C++ Compilers)*』でも、コンパイラーのマイグレーションに関する追加情報が提供されています。

一般には、コンパイラーの新しいバージョンは以前のバージョンと互換性があります。ただし、例外がある場合もあります。例えば、異なる診断メッセージが生成されるかもしれません。コンパイラーをマイグレーションする前に、必ずソース・コードとその他の重要なデータをバックアップしてください。

以下のセクションに、以前のバージョンのコンパイラーで追加された拡張機能がリストされています。新しいバージョンのコンパイラーへのマイグレーションの際に活用してください。

バージョン 12.1 に追加された機能拡張

このセクションでは、バージョン 12.1 でコンパイラーに追加された機能と機能拡張について説明します。

C++11 機能

C++11 は、新しい C++ プログラミング言語標準です。 C++11 は、承認される前には C++0x と呼ばれていました。既存の C++11 の機能に加えて、新たな C++11 の機能が XL C/C++ V12.1 でサポートされています。

注: IBM は、承認される以前は C++0x として知られていた C++11 の、選ばれた機能をサポートします。IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C++11 標準ライブラリーのサポートを含め、 C++11 のすべての機能を IBM が実装し終え

るまで、リリースごとに実装が変更される可能性があります。IBM では、ソース、バイナリー、またはリストおよび他のコンパイラー・インターフェースにおいて、新しい C++11 機能の IBM による以前のリリース実装との互換性を維持する試みは行いません。

XL C/C++ V12.1 には、以下の機能が導入されています。

- 明示的な型変換演算子
- 一般化された定数式
- 参照の縮約 (reference collapsing)
- 右不等号括弧
- 右辺値参照
- スコープ付き列挙型
- 後置戻り型

-qlanglvl=extended0x オプションを使用すると、ほとんどの C++ の機能と、現在サポートされているすべての C++11 の機能を使用可能にすることができます。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qlanglvl』を参照してください。

明示的な型変換演算子

明示的型変換演算子の機能により、`explicit` 関数指定子をユーザー定義の型変換関数の定義に適用することができます。この機能を使用すると、意図しない暗黙の型変換の適用を禁止できるため、あいまいさによるエラーが少ない、より強固なクラスをプログラムできます。

-qlanglvl=explicitconversionoperators オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『明示的型変換演算子 (C++11)』を参照してください。

一般化された定数式

一般化された定数式の機能により、定数式内で使用できる一連の式が拡張されます。定数式とは、コンパイル時に評価できる式のことです。

-qlanglvl=constexpr オプションを使用して、この機能を使用可能にすることができます。

注: XL C/C++ V12.1 では、この機能は C++11 標準で定義されている機能を部分的に実装したものです。

参照の縮約 (reference collapsing)

参照の縮約 (reference collapsing) の機能により、以下のいずれかのコンテキストを使用して参照型に対する参照を形成することができます。

- `decltype` 指定子
- `typedef` 名

- テンプレート型パラメーター

-qlanglvl=referencecollapsing オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『参照の縮約 (reference collapsing) (C++11)』を参照してください。

右不等号括弧

C++ 言語では、閉じ不等号括弧 (>) が 2 つ連続する場合に空白文字で分離しなければなりません。さもなければ、ビット単位の右シフト演算子 (>>) と構文解析されてしまいます。右不等号括弧の機能により、右不等号括弧が連続する場合の空白文字が不要になるため、プログラミングのときに便利です。

-qlanglvl=rightanglebracket オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の 『クラス・テンプレート (C++ のみ)』を参照してください。

右辺値参照

右辺値参照の機能により、引数の値のカテゴリーに基づいて関数を多重定義し、同様の方法で左辺値の特性をテンプレート引数の推定によって検出させることができます。また、右辺値を右辺値参照にバインドして、参照を介して右辺値を変更することもできます。これにより、消失するオブジェクトのリソースを再利用できるプログラミング手法が可能なるため、特にクラス型 (テンプレート・データ構造など) による汎用コードを使用する場合に、ライブラリーのパフォーマンスを向上させることができます。さらに、転送関数を作成する場合に値のカテゴリーを考慮に入れることができます。

-qlanglvl=rvaluereferences オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『右辺値参照の使用 (C++11)』を参照してください。

スコープ付き列挙型

スコープ付き列挙型の機能には以下の利点があります。

- スコープ付き列挙型 (列挙子を列挙型のスコープで宣言します) を宣言できます。
- 列挙子を指定せずに列挙型を宣言できます。列挙子を指定しない列挙型の宣言をフォワード宣言と呼びます。
- 列挙型の基となる型を明示的に指定できます。
- 列挙子の値 (または列挙型のオブジェクト) から整数に変換されなくなり、型の安全性が向上します。

-qlanglvl=scopedenum オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『列挙型』を参照してください。

後置戻り型

後置戻り型の機能は、以下の種類のテンプレートおよび関数を宣言するときに有用です。

- 戻りの型が関数引数の型に依存する関数テンプレートまたはクラス・テンプレートのメンバー関数
- 戻りの型が複雑な関数またはクラスのメンバー関数
- 完全な転送関数

-qlanglvl=autotypededuction オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『後置戻り型 (C++11)』を参照してください。

「*XL C/C++ コンパイラー・リファレンス*」**内の関連情報**

 -qlanglvl

C11 機能

XL C/C++ V12.1 では、C11 の選定された機能に対するサポートが導入されます。C11 は、C プログラミング言語の新しい標準です。

注: IBM は、C11 の選択された機能 (C1X と呼ばれる) をその承認の前にサポートします。 IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C11 標準ライブラリーのサポートを含め、すべての C11 機能を IBM が実装し終えるまで、リリースごとに実装が変更される可能性があります。IBM では、IBM による新規 C11 機能の実装に関し、ソース、バイナリー、リスト作成などのコンパイラー・インターフェースにおいて、以前のリリースとの互換性を維持するための試みは特に行いません。

XL C/C++ V12.1 には、以下の C11 機能が導入されています。

- 無名構造体
- 複素数型の初期化
- 新しい言語レベル - **extc1x**
- `_Noreturn` 関数指定子
- 静的アサーション

無名構造体

この機能により、**extc1x** 言語レベルでの無名構造体の宣言が可能になります。詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『無名構造体』を参照してください。

複素数型の初期化

マクロ CMPLX、CMPLXF、および CMPLXL が標準ヘッダー・ファイル complex.h の中で定義され、**extc1x** 言語レベルでの複素数型の初期化が可能になっています。詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『複素数型の初期化 (C11)』を参照してください。

新しい言語レベル - extc1x

このリリースで、新しいサブオプションが **-qlanglvl** オプションに追加されました。C コンパイラーを使用してコンパイルする場合、**-qlanglvl=extc1x** を使用すると、現在 XL C/C++ によってサポートされている C11 の機能を有効化できます。C++ コンパイラーでコンパイルした場合は、特定の C11 機能も使用可能になります。詳しくは、個々の機能に関する説明のセクションを参照してください。

_Noreturn 関数指定子

_Noreturn 関数指定子は、呼び出し元に戻らない関数を宣言します。この関数指定子を使用して、戻らない独自の関数を定義できます。コンパイラーは、関数から戻る場合に必要な処理を無視することで、よりよいコードを生成できます。詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『_Noreturn 関数指定子』を参照してください。

静的アサーション

静的アサーションが C 言語に追加され、以下の利点がもたらされます。

- ライブラリーが、コンパイル時に一般的な使用エラーを検出できる。
- C 標準ライブラリーの実装が、一般的な使用エラーを検出し、診断できることから、ユーザビリティーが改善される。

静的アサーションを宣言して、重要なプログラム・インバリアントをコンパイル時に検査できます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『_Static_assert 宣言 (C11)』を参照してください。

OpenMP 3.1

XL C/C++ V12.1 は、OpenMP アプリケーション・プログラム・インターフェース・バージョン 3.1 仕様をサポートします。 XL C/C++ 実装は、OpenMP アプリケーション・プログラム・インターフェース・バージョン 3.1 の IBM の解釈に基づいています。

OpenMP 3.1 には、OpenMP 3.0 に対する以下の更新が含まれています。

- 最適化をサポートするために、`task` 構文に `final` および `mergeable` 節が追加されました。
- `taskyield` 構文が追加され、プログラム内でタスクをスイッチできる箇所を、ユーザーが指定できるようになりました。
- 最終タスク領域の特殊化をサポートするために、`omp_in_final` ランタイム・ライブラリー関数が追加されました。

- atomic 構文が拡張され、read、write、および capture 形式が組み込まれました。atomic 構文の既存の形式を適用するための update 節が追加されました。
- 2 つの縮約演算子 min および max が追加されました。
- firstprivate 節での const 修飾型の指定を許可します。
- OpenMP スレッドのプロセッサー間の移動を許可するかどうかを制御するための、OMP_PROC_BIND 環境変数が追加されました。
- OMP_NUM_THREADS 環境変数が拡張され、ネストされた並列領域で使用するスレッド数を指定できるようになりました。

関連情報

- 「*XL C/C++ コンパイラー・リファレンス*」の『OpenMP 環境変数』
- 「*XL C/C++ コンパイラー・リファレンス*」の『並列処理のためのプラグマ・ディレクティブ』
- www.openmp.org

パフォーマンスおよび最適化

XL C/C++ V12.1 での追加の機能および機能拡張は、パフォーマンス調整およびアプリケーション最適化に役立ちます。

コンパイラーの最適化に関するレポート

リスト作成レポートに対して多数の機能拡張が行われて、コンパイラーがコードをどのように最適化したか関する情報がより多く得られるようになっています。この情報を使用して、コンパイラーの最適化機能からさらに多くの利益を得ることができます。これらの機能拡張されたレポートについて詳しくは、『診断レポート』を参照してください。

パフォーマンス・チューニングおよびプログラム最適化について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の 『アプリケーションの最適化』を参照してください。

診断レポート

XL C/C++ V12.1 で追加された新規診断レポートは、コードのパフォーマンスを向上させる機会の識別に役立つことがあります。

HTML 形式でのコンパイラー・レポート

コンパイラーが行うことができた最適化に関して、またいずれの最適化機会を逃したかに関しても、その情報を XML または HTML 形式で取得できるようになりました。この情報を使用して、アプリケーションのチューニングでの、特に高性能アプリケーションのチューニングでのプログラミング労力を減らすことができます。

-qlistfmt オプションおよびそれに関連するサブオプションを使用して、XML または HTML レポートを生成できます。デフォルトでは、このオプションは、内容の種類を指定しなかった場合に、生成可能なすべての内容を生成するようになっています。

既に生成された XML レポートの HTML バージョンを表示するには、**genhtml** ツールを使用できます。このツールの使用方法について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『**genhtml** コマンド』を参照してください。

このレポート、およびレポートの使用法に関して詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『レポートを使用した最適化の機会の診断』を参照してください。

レポートのプロファイル作成の機能拡張

プログラムの分析に役立つ新しいセクションがリスト・ファイルに追加されました。**-qreport** オプションを **-qpdf2** オプションと併用すると、以下のセクションがリスト・ファイルの `PDF Report` というタイトルのセクションに追加されます。

プロファイル・データの関連度 (Relevance of profiling data)
このセクションには、**-qpdf1** フェーズでのプロファイル・データとソース・コードの関連度が示されます。関連度は、0 から 100 の範囲の数値で示されます。値が大きいほどプロファイル・データとソース・コードの関連性が高く、プロファイル・データの使用によるパフォーマンスの向上幅も大きくなります。

欠落しているプロファイル・データ (Missing profiling data)
このセクションには、欠落しているプロファイル・データについての警告メッセージが出力されます。この警告メッセージは、コンパイラーがプロファイル・データを検出しなかった関数それぞれについて出されます。

古いプロファイル・データ (Outdated profiling data)
このセクションに、古いプロファイル・データについての警告メッセージが出力される場合があります。コンパイラーは、**-qpdf1** フェーズの後に変更された関数それぞれについてこの警告メッセージを出します。警告メッセージは、**-qpdf1** フェーズから **-qpdf2** フェーズまでの間に最適化レベルが変更された場合にも出されます。

Profile-Directed Feedback について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『Profile-Directed Feedback の使用』を参照してください。

リスト・ファイルについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー・リスト』を参照してください。

showpdf レポートの機能拡張

現在提供されているブロック・カウンターおよび呼び出しカウンター・プロファイル情報のほかに、**showpdf** ユーティリティーを使用して、キャッシュ・ミス・プロファイルおよび値プロファイル情報も表示できます。値プロファイルおよびキャッシュ・ミス・プロファイル情報は、XML 形式のみで表示できます。ただし、他の種類のプロファイル情報は、すべてテキストまたは XML 形式のいずれかで表示できます。このリリースでは、Profile-Directed Feedback (PDF) 情報が 2 つのファイルに保存されます。1 つは **-qpdf1** フェーズで生成される PDF マップ・ファイルで、もう 1 つは、結果として得られるアプリケーションの実行中に生成される PDF ファイルです。**showpdf** ユーティリティーを実行して、これら 2 つのファイルに保存されている PDF 情報を表示できます。詳しくは、「*XL C/C++ 最適化およびプロ*

グラミング・ガイド」の『showpdf によるプロファイル情報の表示』を参照してください。

新規および拡張された診断オプション

以下の表の項目は、コンパイラー・リスト作成を制御できる、新規または変更されたコンパイラー・オプションおよびディレクティブを説明しています。

ここに記載されている情報は簡単な概要です。上記、およびその他のパフォーマンス関連のコンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『リスト、メッセージ、およびコンパイラー情報』を参照してください。

表 4. リスト関連のコンパイラー・オプションおよびディレクティブ

オプション/ディレクティブ	説明
-qlistfmt	**-qlistfmt** オプションが拡張され、コンパイラーによって行われた最適化、および逃した最適化機会に関する情報を含むレポートを、HTML レポートのほかに XML レポートでも生成できるようになりました。 このオプションのデフォルトの動作は変更されました。現在は、内容の種類を指定しなかった場合に、何も生成しないのではなく、生成可能なすべての内容を生成するようになっています。

組み込み関数

このセクションでは、 V12.1 の新機能である、 組み込み関数の主要なカテゴリーについて説明します。

GCC アトミック・メモリー・アクセス組み込み関数 (IBM 拡張)

このリリースでは、アトミック・メモリー・アクセス用の新しい XL C/C++ 組み込み関数が追加されました。動作は、GNU Compiler Collection (GCC) での動作に対応します。複数のスレッドを使用するプログラムでは、これらの関数を使用して、他のスレッドに干渉することなく、任意のスレッドでデータをアトミックかつ安全に変更できます。

XL C/C++ に用意されている組み込み関数について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー組み込み関数』を参照してください。

コンパイラー・オプションおよびプラグマ・ディレクティブ

このセクションでは、 V12.1 の新規または変更されたコンパイラー・オプションおよびプラグマ・ディレクティブについて説明します。

コマンド行でコンパイラー・オプションを指定することができます。また、アプリケーション・ソース・ファイルに埋め込まれたプラグマ・ディレクティブによりコンパイラーの動作を変更することもできます。 これらのコンパイラー・オプションの詳細説明および使用法に関する情報については、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。

新規または変更されたコンパイラー・オプション

-g **-g** オプションが拡張され、最適化されたプログラムのデバッグを強化するための新しいさまざまなレベルが追加されました。

-qhaltonmsg

以前は C++ コンパイラーのみでサポートされていた **-qhaltonmsg** オプションが XL C によってサポートされるようになりました。これは、指定のエラー・メッセージが生成された場合に、オブジェクト・ファイル、実行可能ファイル、またはアセンブラー・ソース・ファイルを生成する前にコンパイルを停止します。負の形式の **-qnohaltonmsg** も追加されました。

-qinclude

前に指定された **-qinclude** オプションを無視するための、負の形式の **-qnoinclude** が追加されました。

-qinfo **-qinfo=all** により、**als** および **ppt** を除くすべてのグループのすべての診断メッセージが有効化されるようになっています。

-qinitauto

-qinitauto オプションが拡張され、自動変数のワード初期化を実行できるようになりました。

-qkeyword

C++11 新規サブオプション **-q[no]keyword=constexpr** により、`constexpr` キーワードを有効化または無効化できます。

-qlanglvl

以下のサブオプションが追加または更新されています。

C++11 **-qlanglvl=autotypededuction**
このサブオプションにより、auto 型推定に加え、後置戻り型の機能を有効化できるようになりました。

C++ **-qlanglvl=c1xnoreturn**
このサブオプションにより、`_Noreturn` 関数指定子のサポートが有効になります。

C++ **-qlanglvl=complexinit**
このサブオプションにより、複素数型の初期化を使用可能にするかどうかを制御します。

C++ IBM **-qlanglvl=compatrvaluebinding**
このサブオプションは、初期化指定子が不要な場合に、非 const 左辺値参照をユーザー定義型の右辺値にバインドすることをコンパイラーに許可します。

C++11 **-qlanglvl=constexpr**
このサブオプションにより、一般化された定数式の機能が使用可能になり、定数式内で使用できる式が拡張されます。

注: XL C/C++ V12.1 では、この機能は C++11 標準で定義されている機能を部分的に実装したものです。

C++11 **-qlanglvl=explicitconversionoperators**
このサブオプションにより、明示的な型変換演算子の機能が有効に

なります。この機能により、ユーザー定義の型変換関数を通じた意図しない暗黙の型変換を禁止することができます。

C11 **-qlanglvl=extc1x**
このサブオプションにより、現在サポートされているすべての C11 機能と、その他の実装固有の言語拡張が使用可能になります。

C++11 **-qlanglvl=referencecollapsing**
このサブオプションにより、参照の縮約 (reference collapsing) の機能が有効になります。この機能により、`decltype` 指定子、`typedef` 名、またはテンプレート型パラメーターを使用して参照型に対する参照を形成することができます。

C++11 **-qlanglvl=rightanglebracket**
このサブオプションにより、右不等号括弧の機能が有効になります。この機能により、右不等号括弧が連続する場合の空白文字が不要になります。

C++11 **-qlanglvl=rvaluereferences**
このサブオプションにより、右辺値参照機能が有効になります。

C++11 **-qlanglvl=scopedenum**
このサブオプションにより、スコープ付き列挙型の機能が有効になります。この機能により、スコープ付き列挙型や、列挙子を指定しない列挙型を宣言できます。

C++ IBM **-qlanglvl=tempsaslocals**
このサブオプションにより、一時変数の存続期間が延長され、マイグレーションがしやすくなります。

IBM **-qlanglvl=textafterendif**
このサブオプションにより、`#endif` または `#else` の後にテキストを追加できるコンパイラーから IBM XL C/C++ コンパイラーにコードを移植するときに出される警告メッセージが抑制されます。

C++11 の新機能について詳しくは、 25 ページの『C++11 機能』を参照してください。

C11 機能について詳しくは、 28 ページの『C11 機能』を参照してください。

-qlistfmt

-qlistfmt オプションが拡張され、コンパイラーによって行われた最適化、および逃した最適化機会に関する情報を含むレポートを、HTML レポートのほかに XML レポートでも生成できるようになりました。

-qlistfmt のデフォルトの動作が変更されました。このリリースでは、内容の種類を指定しなかった場合に、何も生成しないのではなく、生成可能なすべての内容を生成するようになっています。

-qoptfile

新しいオプション **-qoptfile** は、コンパイルに使用する追加のコマンド行オプションのリストを含むファイルを指定します。

-qpic **-qpic=large** により、大規模 TOC アクセスが有効化され、目次が 64 Kb を超える場合に TOC オーバーフロー条件の発生を予防できます。

-qshowpdf

デフォルト値が **-qnoshowpdf** から **-qshowpdf** に変更されています。

新規または変更されたプラグマ・ディレクティブ

#pragma ibm independent_loop

independent_loop プラグマが追加されました。これは、選択されたループの反復が独立しており、反復を並列に実行できることを明示的に指定します。

#pragma ibm iterations

iterations プラグマが追加されました。これは、選択されたループのループ反復のおおよその回数を指定します。

#pragma ibm max_iterations

max_iterations プラグマが追加されました。これは、選択されたループのループ反復のおおよその最大回数を指定します。

#pragma ibm min_iterations

min_iterations プラグマが追加されました。これは、選択されたループのループ反復のおおよその最小回数を指定します。

#pragma simd_level

simd_level プラグマが追加されました。これは、コンパイラーによる、個々のループを対象としたベクトル命令のコード生成を制御します。

バージョン 11.1 に追加された機能拡張

このセクションでは、バージョン 11.1 でコンパイラーに追加された機能と機能拡張について説明します。

POWER7 プロセッサーのサポート

XL C/C++ for Linux, V11.1 は、POWER7 プロセッサーをサポートします。

POWER7 プロセッサーのサポートで導入された新機能および機能拡張は、以下の 4 つのカテゴリーに分類されます。

- ベクトル・スカラー拡張データ型および組み込み関数
- POWER7 プロセッサー用の MASS ライブラリー
- POWER7 プロセッサー用の組み込み関数
- POWER7 プロセッサー用のコンパイラー・オプション

ベクトル・スカラー拡張データ型および組み込み関数

コンパイラーのこのリリースは、POWER7 プロセッサー内に設定された Vector Scalar eXtension (VSX) 命令セットをサポートします。新しいデータ型、および組み込み関数が導入され、VSX 命令をサポートします。VSX 組み込み関数およびオリジナルの Vector Multimedia eXtension (VMX) 組み込み関数を使用して、アプリケーションで、ベクトル演算を効率的に処理することができます。

VSX データ型および組み込み関数について詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『ベクトル型』、および「*XL C/C++ コンパイラー・リファレンス*」の『ベクトル組み込み関数』を参照してください。

POWER7 プロセッサー用の Mathematical Acceleration Subsystem (MASS) ライブラリー

ベクトル・ライブラリー

ベクトル MASS ライブラリー **libmassvp7.a** には、POWER7 アーキテクチャー用に調整されたベクトル関数が入っています。これらの関数は、32 ビットと 64 ビットのどちらのモードでも使用できます。

以前の Power® プロセッサーをサポートしている関数は、単精度と倍精度のどちらも、POWER7 プロセッサー用に組み込まれています。

以下の新しい関数が、単精度と倍精度の両方の関数グループに追加されています。

- exp2
- exp2m1
- log21p
- log2

ベクトル・ライブラリーについて詳しくは、 『ベクトル・ライブラリーの使用』(「*XL C/C++ 最適化およびプログラミング・ガイド*」内) を参照してください。

SIMD ライブラリー

MASS SIMD ライブラリー **libmass_simdp7.a** には、頻繁に使用される組み込み数学関数の高速セットが含まれています。これらは、対応する標準システム・ライブラリーの関数を上回るパフォーマンスを発揮します。

SIMD ライブラリーについて詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『POWER7 の SIMD ライブラリーの使用』 を参照してください。

POWER7 ハードウェア組み込み機能

以下の POWER7 プロセッサー機能をサポートするために、新しいハードウェア組み込み機能が追加されました。

- 新しい POWER7 プリフェッチ拡張機能およびキャッシュ制御
- 新しい POWER7 ハードウェア命令

詳しくは、 48 ページの『組み込み関数』を参照してください。

POWER7 プロセッサー用の新規コンパイラー・オプション

新しい arch および tune コンパイラー・オプション

-qarch コンパイラー・オプションはコードの生成対象であるプロセッサー・アーキテクチャーを指定します。**-qtune** コンパイラー・オプションは、特定のハードウェア・アーキテクチャーで最も効率よく実行できるように、命令選択、スケジューリング、およびその他のアーキテクチャー依存のパフォーマンス拡張機能をチューニングします。

-qarch=pwr7 は、POWER7 ハードウェア・プラットフォーム上で実行される命令が入っているオブジェクト・コードを生成します。**-qtune=pwr7** を使用すると、POWER7 ハードウェア・プラットフォームに合わせて、最適化が調整されます。

詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qarch』および「*XL C/C++ コンパイラー・リファレンス*」の『-qtune』を参照してください。

C++11 機能

XL C/C++, V11.1 では、C++ プログラミング言語の新しい標準である C++11 の、選定された機能に対するサポートが導入されています。

注: IBM は、承認される以前は C++0x として知られていた C++11 の、選ばれた機能をサポートします。IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C++11 標準ライブラリーのサポートを含め、 C++11 のすべての機能を IBM が実装し終えるまで、リリースごとに実装が変更される可能性があります。IBM では、ソース、バイナリー、またはリストおよび他のコンパイラー・インターフェースにおいて、新しい C++11 機能の IBM による以前のリリース実装との互換性を維持する試みは行いません。

XL C/C++ V11.1 には、以下の機能が導入されています。

- auto 型推定
- C99 `long long`
- C++11 で採用されている C99 プリプロセッサー機能
- decltype
- 委任コンストラクター
- 明示的インスタンス生成宣言
- 拡張フレンド宣言
- インライン名前空間定義
- 静的アサーション
- 可変数引数テンプレート

-qlanglvl=extended0x オプションを使用すると、ほとんどの C++ の機能と、現在サポートされているすべての C++11 の機能を使用可能にすることができます。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qlanglvl』を参照してください。

auto 型推定

auto 型推定機能を使用すると、変数の宣言時に型を指定する必要がなくなります。これは、auto 型推定では、自動変数の型推定のタスクを、その初期化指定子の式の型から、コンパイラーに委任するためです。

-qlanglvl=autotypededuction オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の 『auto 型指定子 (C++11)』を参照してください。

C99 long long

C++ コンパイラーは、C99 long long 機能を使用できます。これにより、C 言語と C++ 言語の間のソースの互換性が向上します。

-qlanglvl=c99longlong オプションを使用して、C99 long long 機能を使用可能にすることができます。

IBM この機能を使用可能にした後、u または U を含む接尾部を持たない 10 進整数リテラルを long long int 型で表現できない場合には、-qlanglvl=[no]extendedintegersafe オプションを指定することにより、unsigned long long int 型を使用してそのリテラルを表現するかどうかを決定できます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『整数リテラル』を参照してください。

C++11 で採用されている C99 プリプロセッサー機能

C++11 で採用されたいくつかの C99 プリプロセッサー機能では、C コンパイラーと C++ コンパイラーは、より共通性の高いプリプロセッサー・インターフェースを提供します。これにより、C ソース・ファイルを C++ コンパイラーに容易に移植でき、C プリプロセッサーと C++ プリプロセッサーの間のセマンティックの違いを除去できます。また、プリプロセッサーの互換性の問題が生じたり、プリプロセッサーが異なる動作をしたりするのを回避できます。

-qlanglvl=c99preprocessor オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『C++0x に採用された C99 プリプロセッサー機能 (C++11)』を参照してください。

decltype

decltype 機能を使用すると、型に依存する可能性がある式の結果の型に基づく型を取得することができます。

-qlanglvl=decltype オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『decltype(expression) 型指定子 (C++11)』を参照してください。

委任コンストラクター

委任コンストラクター機能を使用すると、共通の初期化を 1 つのコンストラクターに集中させることができます。これにより、プログラムが読みやすく、また保守しやすくなります。

-qlanglvl=delegatingctors オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『委任コンストラクター (C++11)』を参照してください。

明示的インスタンス生成宣言

明示的インスタンス生成宣言機能を使用すると、テンプレートの特殊化、またはそのメンバーの暗黙のインスタンス生成を抑制することができます。

個別のサブオプション **-qlanglvl=externtemplate** またはグループ・オプション **-qlanglvl=extended** または **-qlanglvl=extended0x** を使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『明示的インスタンス生成 (C++ のみ)』を参照してください。

拡張フレンド宣言

拡張フレンド宣言機能は、フレンド宣言を支配する構文規則を、以下のように緩和します。

- テンプレート・パラメーター、`typedef` 名、および基本型をフレンドとして宣言できる。
- C++11 では、フレンド宣言のコンテキスト内のクラス・キーが必要なくなる。

-qlanglvl=extendedfriend オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『フレンド (C++ のみ)』を参照してください。

インライン名前空間定義

インライン名前空間定義は、初期 `inline`キーワードを使用する名前空間定義です。インライン名前空間のメンバーを、その名前空間を含むエンクロージング名前空間に属しているかのように定義および特殊化することができます。

-qlanglvl=inlinenamespace オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『インライン名前空間定義 (C++11)』を参照してください。

静的アサーション

静的アサーション機能には、以下の利点があります。

- ライブラリーが、コンパイル時に一般的な使用エラーを検出できる。
- C++ 標準ライブラリーの実装が、一般的な使用エラーを検出し、診断できることから、ユーザビリティーが改善される。

コンパイル時に、static_assert 宣言を使用して、重要なプログラム・インバリアントを検査できます。

-qlanglvl=static_assert オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『static_assert 宣言 (C++11)』を参照してください。

可変数引数テンプレート

可変数引数テンプレート機能を使用すると、任意の数 (ゼロを含む) のパラメーターを持つクラス・テンプレートまたは関数テンプレートを定義できます。

-qlanglvl=variadic[templates] オプションを使用して、この機能を使用可能にすることができます。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『可変数引数テンプレート (C++11)』を参照してください。

「*XL C/C++ コンパイラー・リファレンス*」**内の関連情報**

 -qlanglvl

パフォーマンスおよび最適化

追加の機能および機能拡張を、パフォーマンスの調整とアプリケーションの最適化に役立てることができます。

-qpdf の機能拡張

-qpdf オプションの使用は、次の 2 つのステップに分かれています。まず、**-qpdf1** オプションを使用してプログラムをコンパイルします。プログラムを標準的なデータ・セットを使用して実行し、プロファイル作成データを生成します。次に、**-qpdf2** オプションを使用してプログラムをもう一度コンパイルし、プロファイル作成データに基づいてプログラムを最適化します。

以前のリリースでは、ソース・ファイルを変更し、**-qpdf2** オプションを使用してコンパイルすると、コンパイルはエラーを起こして停止しました。XL C/C++ for Linux, V11.1 では、コンパイラーは警告のリストを出しますが、コンパイルは停止しません。これにより、ソース・ファイルの変更後もプロファイル・データを継続して使用できます。

いくつかの新しいサブオプションが **-qpdf** オプションに追加されています。これらの新しいサブオプションを使用すると、パフォーマンスの改善をより細かく制御でき、**-qpdf** を拡張して、マルチパス・プロファイル、キャッシュ・ミス・プロファイル、および拡張された値のプロファイルをサポートできます。

新規 **-qpdf** サブオプションは、以下のとおりです。

level マルチパス・プロファイル、単一パス・プロファイル、キャッシュ・ミス・プロファイル、値プロファイル、ブロック・カウンター・プロファイル、および呼び出しカウンター・プロファイルをサポートしま

す。**-qpdf1=level=0|1|2** を指定してプログラムをコンパイルすると、結果として得られるアプリケーションによって生成されるプロファイル情報の種類を指定できます。

exename

-o オプションによって指定された出力ファイル名に従って、生成する PDF ファイルの名前を指定します。

defname

PDF ファイルをデフォルトのファイル名に戻します。

これらのサブオプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qpdf1、-qpdf2』を参照してください。

コンパイラーの最適化に関するレポート

リスト作成レポートに対して多数の機能拡張が行われて、コンパイラーによるコードの最適化方法に関する情報がより多く得られるようになっています。この情報を使用して、コンパイラーの最適化機能からさらに多くの利益を得ることができます。これらの機能拡張されたレポートについて詳しくは、42 ページの『新規診断レポート』を参照してください。

パフォーマンス関連のコンパイラー・オプションおよびディレクティブ

次の表の項目は、新規または変更されたコンパイラー・オプションおよびディレクティブを説明しています。

ここで提示されている情報は、簡単な概要です。上記のオプション、ディレクティブ、およびその他のパフォーマンス関連のコンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『最適化およびチューニング・オプション』を参照してください。

表 5. パフォーマンス関連のコンパイラー・オプションおよびディレクティブ

-qinline=level=*number*	デフォルト値の 5 を基準としたインライン化の相対値に関する指示をコンパイラーに伝えるために、-qinline に新しいオプションが追加されました。*number* は、0 から 10 の間の整数値の範囲であり、使用するインライン化のレベルを示します。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qinline』を参照してください。
-qpdf	**-qpdf** は、さまざまな PDF の最適化を制御する際に、より柔軟な制御を行えるようにするサブオプションを提供します。詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qpdf1、 -qpdf2』セクションを参照してください。
-qprefetch	コードのパフォーマンスを改善する機会がある場所に、プリフェッチ命令を自動的に挿入するための新規の機能拡張が **-qprefetch** に追加されています: `-qprefetch=assistthread`。 詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qprefetch』を参照してください。

パフォーマンス・チューニングおよびプログラム最適化について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の 『アプリケーションの最適化』を参照してください。

新規診断レポート

新規診断レポートは、コードのパフォーマンスを向上させる機会を識別するのに役立つ可能性があります。

XML 形式でのコンパイラー・レポート

コンパイラーが行うことができた最適化に関してや、いずれの最適化機会を逃したかに関しても、その情報を XML 形式で取得できるようになりました。この情報を使用して、アプリケーションのチューニングでの、特に高性能アプリケーションのチューニングでのプログラミング労力を減らすことができます。

コンパイラーからの情報は、XML 1.0 形式で生成されます。このレポートは XML スキーマによって定義され、ユーザーが結果の読み取りと分析のために作成できるツールで簡単に取り込むことができます。レポートを人間が読むことができる形式にレンダリングするために、スタイルシート `xlstyle.xsl` が用意されており、この形式は、XSLT をサポートするブラウザーで誰でも読み取ることができます。

本リリースでは、以下の 4 つの最適化カテゴリーがレポートで使用可能です。

- インライン化
- ループ変換
- データ再編成
- Profile-Directed Feedback 情報

新しい **-qlistfmt** オプションおよびそれに関連するサブオプションを使用して、新規 XML 1.0 レポートを生成できます。

このレポート、およびレポートの使用法に関して詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『レポートを使用した最適化の機会の診断』を参照してください。

レポートのプロファイル作成の機能拡張

プログラムの分析に役立つ新しいセクションがリスト・ファイルに追加されました。**-qreport** オプションを **-qpdf2** オプションと併用すると、以下のセクションがリスト・ファイルの `PDF Report` というタイトルのセクションに追加されます。

Loop iteration count
最も頻繁なループの繰り返し数と平均の繰り返し数 (入力データの指定されたセットを対象) が、プログラム内の大部分のループについて算出されます。この情報は、プログラムが最適化レベル -O5 でコンパイルされているときのみ使用できます。

Block and call count
レポートのこのセクションには、プログラムの呼び出し構造と、呼び出された関数ごとの各実行数が含まれます。また、各関数のブロック情報も含まれ

ます。非ユーザー定義関数の場合、実行数のみが含まれます。合計ブロックと呼び出し範囲、および実行カウントの降順で示したユーザー関数リストが、このレポート・セクションの末尾に出力されます。さらに、リスト・ファイル内の疑似コードの各ブロックでは、冒頭にブロック・カウント情報が出力されます。

Cache miss
レポートのこのセクションは、単一テーブルで印刷されます。ここでは、特定の関数のキャッシュ・ミスの数と、関数に関する追加情報、例えば、キャッシュ・レベル、キャッシュ・ミス率、行番号、ファイル名、およびメモリー参照などが報告されます。

注: このレポートを作成するには、**-qpdf1=level=2** オプションを使用する必要があります。
また、**PDF_PM_EVENT** 環境変数を実行時に使用して、プロファイルを作成するキャッシュ・レベルを選択することもできます。

Profile-Directed Feedback に関して詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の『Profile-Directed Feedback の使用』を参照してください。

リスト・ファイルについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー・リスト』を参照してください。

データ再編成のレポート

コンパイラーは、リスト・ファイル内に以下の情報を生成できます。

- データ再編成 (コンパイラーによってプログラム変数データがどのように再編成されたかのサマリー)
- コンパイラーによって挿入されたデータ・プリフェッチ命令の位置

データ再編成情報を生成するには、**-qreport** とともに最適化レベル **-qipa=level=2** または **-O5** を指定します。IPA リンク・パス時に、プログラム変数データ用のデータ再編成メッセージが、リスト・ファイルのデータ再編成セクションに、ラベル `DATA REORGANIZATION SECTION` とともに追加されます。再編成には次のものが含まれます。

- 配列分割
- 配列転置
- メモリー割り振りのマージ
- 配列インターリービング
- 配列合体

データ・プリフェッチ挿入ロケーションに関する情報を生成するには、最適化レベル **-qhot** を使用するか、**-qhot** を暗黙指定する何か他のオプションを **-qreport** とともに使用します。この情報は、リスト・ファイルの `LOOP TRANSFORMATION SECTION` に表示されます。

追加ループ分析

新しいサブオプションが **-qhot** に追加され、より積極的なループ分析が加わりました。**-qhot=level=2** が **-qsmp** および **-qreport** とともに使用されることにより、リス

ト・ファイルの LOOP TRANSFORMATION SECTION に、実行された積極的なループ分析の対象となるループ・ネストに関する情報が追加されます。この情報は、**-qlistfmt** オプションで作成される XML リスト・ファイルにも表示することができます。

新規および拡張された診断オプション

以下の表の項目は、コンパイラー・リスト作成を制御できる、新規または変更されたコンパイラー・オプションおよびディレクティブを説明しています。

ここで提示されている情報は、簡単な概要です。上記、およびその他のパフォーマンス関連のコンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『リスト、メッセージ、およびコンパイラー情報』を参照してください。

表 6. リスト関連のコンパイラー・オプションおよびディレクティブ

オプション/ディレクティブ	説明
-qlistfmt	コンパイラーによって行われた最適化、および逃した最適化機会に関する情報が含まれているレポートを、XML 1.0 形式で生成します。このレポートには、インライン化、ループ変換、データ再編成、および Profile-Directed Feedback に関する情報が含まれています。
-qreport	**-qpdf2** を指定して使用される場合、リストに PDF report セクションが含まれるようになりました。 **-qipa=level=2** または **-O5** を指定して使用される場合、リスト・ファイルに、もう 1 つの新規セクション DATA REORGANIZATION が含まれます。
-qskipsrc	コンパイラーによってスキップされたソース・ステートメントが、リスト・ファイルの SOURCE セクションに表示されるかどうかを決定します。

使用状況のトラッキングおよびレポート作成ツール

使用状況のトラッキングおよびレポート作成機能は、組織内でのコンパイラーの使用状況を追跡するための、軽量で単純なメカニズムです。これは、デフォルトでは使用不可に設定されています。この機能を使用して、組織内でのコンパイラーの使用がコンパイラー・ライセンス資格を超過しているかどうかを検出できます。

使用状況のトラッキングが使用可能に設定されている場合、コンパイラーの各呼び出しは、コンパイラー使用状況ファイルに記録されます。使用状況レポート作成ツールを実行して、これらのファイルの 1 つ以上からレポートを生成し、組織内でのコンパイラーの全体的な使用状況の実態を把握できます。**urt** コマンドを使用すると、レポートの生成方法を制御できます。特に、このレポートは、コンパイラーを使用している同時ユーザーの数を示します。

使用状況のトラッキングおよびレポート作成機能はセットアップと管理が容易であり、使用状況をトラッキングしても、コンパイラーの使用またはパフォーマンスに影響が出ません。

使用状況のトラッキングおよびレポート作成機能について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『コンパイラー使用状況のトラッキングとレポート作成』を参照してください。

新規または変更されたコンパイラー・オプションおよびディレクティブ

このセクションでは、XL C/C++, V11.1 の新規または変更されたコンパイラー・オプションおよびディレクティブについて説明します。

コマンド行でコンパイラー・オプションを指定することができます。また、アプリケーション・ソース・ファイルに埋め込まれたプラグマ・ディレクティブによりコンパイラーの動作を変更することもできます。 これらのコンパイラー・オプションの詳細説明および使用法に関する情報については、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。

表 7. 新規または変更されたコンパイラー・オプションおよびディレクティブ

オプションまたはディレクティブ	説明
-qarch	新しいサブオプションが **-qarch** に追加されており、**-qarch=pwr7** を指定すると、POWER7 ハードウェア・プラットフォーム上で実行する命令が入ったオブジェクト・コードが生成されます。
-qassert	**-qassert** は、XL C/C++ 用の新しいオプションです。これを使用して、最適化を微調整するのに役立つファイルの特性に関する情報を得ることができます。
-qfunctrace	コンパイル単位内または特定の関数リスト内のみの関数の入り口点と出口点をトレースします。
-qhot	新しいサブオプションが **-qhot** 用に追加されました。 **-qhot** コンパイラー・オプションは、手動調整に対する強力な代替物で、ループおよび配列言語を最適化することができます。 **-qhot=fastmath** オプションを使用すると、**-qstrict=nolibrary** が有効になっているときのみ、数学ルーチンを、XLOPT ライブラリーから取得可能な数学ルーチンに置き換えることができます。**-qhot=nofastmath** を指定すると、XLOPT ライブラリーを使用した数学ルーチンの置き換えは無効になります。-qhot=fastmath は、ホット・レベルに関係なく、-qhot が指定されていると、デフォルトで有効になります。
-qinline	パフォーマンスを向上するため、関数に対する呼び出しを生成するのではなく、関数をインライン化します。
-qkeepinlines	新しいサブオプション **exports** が、**-qkeepinlines** オプションに追加されました。**-qkeepinlines=exports** を使用して、前のバージョンのコンパイラーを使用してコンパイルされた共有オブジェクト・ファイルから、シンボルとシンボルの定義のリストをコンパイラーが保持するようにできます。

表 7. 新規または変更されたコンパイラー・オプションおよびディレクティブ (続き)

オプションまたはディレクティブ	説明
-qlanglvl	C++11 新しいサブオプションが **-qlanglvl** に追加されました。 • **-qlanglvl=autotypedededuction**: auto 型推定機能を使用可能に設定するかどうかを制御します。この機能を使用して、自動変数の型推定のタスクを、その初期化指定子の式の型から、コンパイラーに委任することができます。 • **-qlanglvl=c99longlong**: C99 `long long` 機能を使用可能に設定するかどうかを制御します。この機能は C と C++ 言語間のソース互換性を向上させます。 • **-qlanglvl=c99preprocessor**: C++11 で採用されている C99 プリプロセッサー機能を使用可能に設定するかどうかを制御します。この機能を使用して、C および C++ コンパイラーに、より共通性の高いプリプロセッサー・インターフェースを提供することができます。 • **-qlanglvl=decltype**: decltype 機能を使用可能に設定するかどうかを制御します。この機能を使用して、型に依存する可能性がある式の結果の型を基にした型を取得することができます。 • **-qlanglvl=delegatingctors**: 委任コンストラクター機能を使用可能に設定するかどうかを制御します。この機能を使用して、共通の初期化を 1 つのコンストラクターにまとめることができます。 • **-qlanglvl=extendedfriend**: 拡張フレンド宣言機能を使用可能に設定するかどうかを制御します。この機能を使用して、追加の非関数フレンド宣言形式を受け入れることができます。 • IBM **-qlanglvl=extendedintegersafe**: `u` または `U` を含む接尾部を持たず、`long long int` 型で表現できない 10 進整数リテラルの型として、`unsigned long long int` を使用できるかどうかを制御します。このオプションは、**-qlanglvl=c99longlong** オプションが指定されている場合にのみ有効になります。 • **-qlanglvl=externtemplate**: 明示的インスタンス生成宣言機能を使用可能に設定するかどうかを制御します。この機能を使用して、テンプレートの特殊化、またはそのメンバーの暗黙のインスタンス生成を抑制することができます。 • **-qlanglvl=inlinenamespace**: インライン名前空間定義機能を使用可能に設定するかどうかを制御します。この機能を使用して、インライン名前空間のメンバーを、エンクロージング名前空間のメンバーでもあるかのように定義および特殊化することができます。 • **-qlanglvl=static_assert**: 静的アサーション機能を使用可能に設定するかどうかを制御します。この機能を使用して、失敗時にサーバー・エラー・メッセージが発行される先のコンパイル時アサーションを作成できます。 • **-qlanglvl=variadic[templates]**: 可変数引数テンプレート機能を使用可能に設定するかどうかを制御します。この機能を使用して、任意の数 (ゼロを含む) のパラメーターを持つクラス・テンプレートまたは関数テンプレートを定義できます。

表 7. 新規または変更されたコンパイラー・オプションおよびディレクティブ (続き)

オプションまたはディレクティブ	説明
-qlibmpi	Message Passing Interface (MPI) 関数の既知の動作を基にして、コードを調整します。
-qlistfmt	インライン化、ループ変換、Profile-Directed Feedback、およびデータ再編成に関して、コンパイラーによって行われたいくつかの最適化の情報、およびいくつかの逃した最適化の機会の情報が含まれているレポートを、XML 1.0 形式で生成します。
-qpdf1、qpdf2	新しいサブオプションが **-qpdf1,-qpdf2** に追加されました。
-qprefetch	新しいサブオプションが **-qprefetch** に追加されました。高いキャッシュ・ミス率を生じるアプリケーションを処理する場合、**-qprefetch=assistthread** を使用して、データ・プリフェッチ用の支援スレッドを活用することができます。
-qrestrict (C のみ)	**-qrestrict** を使用して、すでに関数仮パラメーター・ポインターによってアドレス指定された同じメモリーに、他のポインターがアクセスできないことをコンパイラーに示すことができます。
-qsaveopt|-qnosaveopt	既存の **-qsaveopt** オプションが、ユーザーの構成ファイル名と構成ファイル内で指定されたオプションも含むように拡張されました。
-qstackprotect	スタックを上書きまたは破損する、悪意のあるコードまたはプログラム・エラーから、アプリケーションを保護します。
-qstaticlink	共有ランタイム・ライブラリーおよび非共有ランタイム・ライブラリーをアプリケーションにリンクさせる方法を制御できます。
-qstrict	新しいサブオプションが **-qstrict** オプションに追加されることで、最適化の制御、および厳密なプログラム・セマンティクスに違反する変換の制御が向上しました。 **-qstrict=vectorprecision** は、ベクトル化した繰り返しによって、ベクトル化しない繰り返しと異なる結果が生成される可能性がある場合、ループでのベクトル化を使用不可にします。
-qtune	新しいサブオプションが **-qtune** に追加されました。 **-qtune=pwr7** を指定すると、POWER7 ハードウェア・プラットフォームに合わせて最適化が調整されます。

表 8. 非推奨のディレクティブおよびオプション

オプションまたはディレクティブ	説明
-Q	このオプションは非推奨で、**-qinline** に置き換えられます。
-qenablevmx	このオプションは非推奨で、**-qsimd=auto** オプションに置き換えられます。
-qhot=simd | nosimd	**-qhot=simd | nosimd** は非推奨で、今後のリリースで除去される可能性があります。**-qsimd** を使用できます。
-qinfo=private	**-qinfo=private** は非推奨で、**-qreport** に置き換えられます。
-qinfo=reduction	**-qinfo=reduction** は非推奨で、**-qreport** に置き換えられます。

表 8. 非推奨のディレクティブおよびオプション (続き)

オプションまたはディレクティブ	説明
-qipa=inline | noinline	**-qipa=inline | noinline** は非推奨で、今後のリリースで除去される可能性があります。**-qinline** を使用できます。
-qipa=clonearch | noclonearch	**-qipa=clonearch | noclonearch** は、現在サポートされていません。**-qtune=balanced** を使用できます。
-qipa=clonearch | noclonearch	**-qipa=cloneproc | nocloneproc** は、現在サポートされていません。**-qtune=balanced** を使用できます。

組み込み関数

このセクションでは、XL C/C++ V11.1 における新規の組み込み関数をリストします。

XL C/C++ に用意されている組み込み関数について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー組み込み関数』を参照してください。

VSX 組み込み関数

POWER7 プロセッサー用に、Vector Scalar eXtension (VSX) が新規に追加されました。

VSX 組み込み関数について詳しくは、 『ベクトル組み込み関数』を参照してください。

POWER7 プリフェッチ拡張機能およびキャッシュ制御

POWER7 プロセッサーには、ストア・ストリーム・プリフェッチおよびプリフェッチ深さ制御をサポートするキャッシュ制御およびストリーム・プリフェッチ拡張機能があります。 XL C/C++ は、以下の新規組み込み関数を提供して、これらの命令にプログラマーが直接アクセスできるようにしています。

- `__protected_stream_stride`
- `__transient_protected_stream_count_depth`
- `__unlimited_protected_stream_depth`
- `__transient_unlimited_protected_stream_depth`
- `__partial_dcbt`
- `__dcbtt`
- `__dcbtstt`
- `__dcbflp`

コンパイラーはコードを最適化するときに、組み込み関数を自動的に挿入できます。**-qnoprefetch** を使用すると、これらの命令が自動的に使用されないようにすることができます。

ディレクティブについて詳しくは、 「*XL C/C++ コンパイラー・リファレンス*」の『組み込み関数』を参照してください。

POWER7 ハードウェア組み込み関数

このリリースでは、それぞれの新規 POWER7 ハードウェア命令に対応する新しい XL C/C++ 組み込み関数が追加されました。それらの関数を使用して、コード内の特定のハードウェア命令を直接操作し、アプリケーションのパフォーマンスを向上させることができます。

- `__bpermd`
- `__cbcdtd`
- `__cdtbcd`
- `__load8r`
- `__store8r`
- `__divde`
- `__divdeu`
- `__cmpb`
- `__divwe`
- `__divweu`
- `__addg6s`

変換関数

これらの新規関数は、デクレット (Declet) とバイナリー・コード化 10 進数の間の変換を行います。

- `__cbcdtd`
- `__cdtbcd`

比較関数

この新規関数は、バイトを比較します。

- `__cmpb`

10 進浮動小数点関数

この新機関数は、sixes (6 つからなるグループ) を追加および生成します。

- `__addg6s`

バージョン 10.1 に追加された機能拡張

このセクションでは、バージョン 10.1 でコンパイラーに追加された機能と機能拡張について説明します。

C++11 機能

XL C/C++ V10.1 では、 C++ プログラミング言語、具体的には、承認前に C++0x としても知られていた C++11 の標準の新規バージョンのサポートが導入されていま

す。リリース時点で標準は正式には採用されていませんでしたが、当社では、その機能の一部のサポートを開始していました。

注: IBM は、承認される以前は C++0x として知られていた C++11 の、選ばれた機能をサポートします。IBM は、この標準の機能の開発および実装を継続します。この言語レベルの実装は、IBM による標準の解釈に基づいています。新しい C++11 標準ライブラリーのサポートを含め、 C++11 のすべての機能を IBM が実装し終えるまで、リリースごとに実装が変更される可能性があります。IBM では、ソース、バイナリー、またはリストおよび他のコンパイラー・インターフェースにおいて、新しい C++11 機能の IBM による以前のリリース実装との互換性を維持する試みは行いません。

特に、このリリースでは、

- 新規の言語レベルを追加します。
- long long データ型を使用した、算術変換用の新しい整数上位変換規則が導入されています。
- C++ プリプロセッサーは、現在 C99 フィーチャーをサポートします

新しい言語レベル - extended0x

C++ コンパイラーを呼び出すときデフォルトの **-qlanglvl** コンパイラー・オプションは **extended** のままです。

このリリースで、新しいサブオプションが **-qlanglvl** オプションに追加されました。**-qlanglvl=extended0x** を使用すると、ユーザーはすべての機能の初期の実装を試行でき、これは XL C/C++ により現在サポートされている C++11 の機能です。

C++ での C99 long long

XL C/C++, V10.1 のこのリリースでは、整数リテラル・データ型で特定の算術演算を実行すると、コンパイラーの動作が変化します。特に、整数の上位変換規則は変更されました。

以前は、C++ (および C89 への拡張として) では、**-qlonglong** を指定してコンパイルするとき、サフィックスなしの整数リテラルは、適合するこのリスト内の最初のタイプにレベル上げされます。

```
int
long int
unsigned long int
long long int
unsigned long long
```

このリリースから、および **-qlanglvl=extended0x** を指定してコンパイルするとき、コンパイラーはサフィックスなしの整数リテラルを、適合するこのリスト内の最初のタイプにレベル上げします。

```
int
long int
long long int
```

```
unsigned long long
```

注: C コンパイラーの C99 標準の弊社の実装のように、ある値が `long long` タイプに適合できないが、`unsigned long long` タイプに適合できる場合、C++ では `long long` から `unsigned long long` に上位変換が可能です。この場合、メッセージが生成されます。

マクロ `__C99_LLONG` は C99 との互換性のために追加されています。このマクロは、**-qlanglvl=extended0x** を指定すると、1 に定義され、それ以外の場合は未定義です。

For more information, see "Integral and floating-point promotions" in the *XL C/C++ ランゲージ・リファレンス*.

プリプロセッサーの変更

以下の変更によって、C++ プリプロセッサーは、C から C++ へのコードの移植を、より簡単に行うことができます。

- 通常のストリング・リテラルは、広幅ストリング・リテラルを使用して連結できるようになりました。
- `#line <integer>` プリプロセッサー・ディレクティブの上限が大きくなりました。これは、C++ の場合、32767 から 2147483647 に増えました。
- C++ が `_Pragma` 演算子をサポートするようになりました。
- 以下のマクロは、C だけでなく C++ にも適用されるようになりました。
  - __C99_MACRO_WITH_VA_ARGS (**-qlanglvl=extended** を指定しても使用可能)
  - __C99_MAX_LINE_NUMBER (**-qlanglvl=extended** を指定しても使用可能)
  - __C99_PRAGMA_OPERATOR
  - __C99_MIXED_STRING_CONCAT

注: 注記された場合以外、C++ プリプロセッサーの変更は、**-qlanglvl=extended0x** を指定してコンパイル場合にのみ使用可能です。

XL C/C++ でサポートされている言語標準について詳しくは、 「*XL C/C++ ランゲージ・リファレンス*」の 『言語レベルおよび拡張』 を参照してください。

その他の XL C/C++ 言語関連の更新

このセクションでは、バージョン 10.1 で導入された言語関連の変更について説明します。

ベクトル・データ型

ベクトル・データ型は、以下の基本データ型で使用できる演算子の一部を使用できるようになりました。

- 単項演算子
- 2 項演算子
- 関係演算子

OpenMP 3.0

IBM XL C/C++ for Linux, V10.1 では、OpenMP API バージョン 3.0 仕様がサポートされます。 XL C/C++ 実装は、OpenMP アプリケーション・プログラム・インターフェース・ドラフト 3.0 パブリック・コメントの IBM の解釈に基づきます。

バージョン 2.5 と、バージョン 3.0 との主な違いは以下のとおりです。

- タスク・レベルの並列化の追加。新規の OpenMP 構成体 `TASK` および `TASKWAIT` は、既存の OpenMP 構成体では適切ではなかったポインター追跡または再帰的アルゴリズムなどの、不規則アルゴリズムを並列化する能力をユーザーに与えます。
- `for` ループには、`signed int` と同様に `unsigned int` の *var* 値、および `pointer` 型を入れることができるようになりました。
- スタック・サイズの制御。OMP ランタイム・ライブラリーによって作成されたスレッドのスタックのサイズを、新しい環境変数 `OMP_STACKSIZE` を使用して制御できるようになりました。
- ユーザーは、新しい環境変数 `OMP_WAIT_POLICY` および `OMP_SET_POLICY` を使用して、待機スレッドの望ましい動作に関してヒントを与えることができるようになりました。
- ストレージの再使用。`PRIVATE` 節のいくつかの制限は除去されました。並列構成体の `reduction` 節に表示されるリスト項目は、ワーク・シェアリング構成体の `private` 節でも表示できるように成りました。
- スケジューリング。新規の `SCHEDULE` 属性では、`auto` がコンパイラーおよび実行時システムにスケジューリングの制御を許可します。
- `STATIC` スケジュールを指定した連続ループ構成体で `nowait` を使用できるように成りました。
- ネスト・サポート - `DO`、`FOR`、`PARALLEL FOR`、 および `PARALLEL DO` ディレクティブに `COLLAPSE` 節が追加されて、完全なループ・ネスト並列化が可能になりました。これは、1 つのネスト内の複数のループを並列処理できることを意味します。
- `THREADPRIVATE` ディレクティブは、ファイルおよびブロック・スコープに加えて、クラス・スコープに変数を適用できるように成りました。
- ランダム・アクセス・イテレーターを指定した、これらを含む標準フォームのイテレーター・ループの並列化。

詳細については、以下を参照してください。

- 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『OpenMP ディレクティブの使用』
- www.openmp.org

パフォーマンスおよび最適化

XL C/C++, V10.1 には、アプリケーションのパフォーマンス・チューニングおよび最適化を支援するフィーチャーや機能拡張が組み込まれています。

-qstrict の機能拡張

多くのサブオプションが **-qstrict** オプションに追加されることで、最適化の制御、および厳密なプログラム・セマンティクスに違反するトランスフォーメーションのよりきめの細かい制御が可能になりました。前のリリースでは、**-qstrict** オプションは、厳密なプログラム・セマンティクスに違反するトランスフォーメーションをすべて使用不可にしていました。これは、サブオプションなしで **-qstrict** を使用した場合の動作であることには変わりありません。 同様に、以前のリリースでは、**-qnostrict** はプログラムのセマンティクスの変更が可能なトランスフォーメーションを許可しました。高水準の最適化では厳密なプログラム・セマンティクスを緩和することが必要になる場合があるため、サブオプションが追加されることにより、選択した規則が緩和されて、すべてのセマンティクス検査をオフにすることなく、高速コードの特定の利点が得られます。

16 個の新規サブオプションを別々に使用することも、サブオプションのグループを使用することもできます。以下に、サブオプション・グループをリストします。

all
: 他のサブオプションにより制御されるものを含む、すべてのセマンティクス変更のトランスフォーメーションを使用不可にします。

ieeefp
: 個々のオペレーションが IEEE 754 セマンティクスに準拠しているかどうかを制御します。

order
: プログラム言語セマンティクスに違反するような方法で、個々のオペレーションを再配列できるかどうかを制御します。

precision
: プログラムの結果の精度に影響を与える可能性のある、最適化および変換を制御します。

exceptions
: プログラムにより生成される実行時例外に影響を与える可能性のある、最適化および変換を制御します。

これらのサブオプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『-qstrict』を参照してください。

パフォーマンス関連のコンパイラー・オプションおよびディレクティブ

次の表の項目は、新規または変更されたコンパイラー・オプションおよびディレクティブを説明しています。

ここで提示されている情報は、簡単な概要です。上記、およびその他のパフォーマンス関連のコンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『最適化およびチューニング・オプション』を参照してください。

表 9. パフォーマンス関連のコンパイラー・オプションおよびディレクティブ

オプション/ディレクティブ	説明
-qstrict	多くの新しいサブオプションが追加され、一定のパフォーマンスの利点を活用できるよう、プログラム・セマンティクス規則の緩和についてユーザーがよりよく制御できるようにしています。
-qfloat	一部の **-qfloat** サブオプションは、 **-qstrict** の新しいサブオプションによって影響を受けます。
-qreport	リスト表示には、各ループ用に作成されたストリームの数、および非ストライド・ワン参照のため SIMD ベクトル化ができないループについての情報が入るようになりました。ユーザーはこの情報を使用して、アプリケーションのパフォーマンスを改善できます。
-qsmp	**-qsmp=omp** が有効になっているとき、OpenMP API 3.0 の追加機能が使用できるようになりました。詳しくは、52 ページの『OpenMP 3.0』を参照してください。

パフォーマンス・チューニングおよびプログラム最適化について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の 『アプリケーションの最適化』を参照してください。

コンパイラー・オプションおよびプラグマ・ディレクティブ

このセクションでは、XL C/C++, V10.1 の新規または変更されたコンパイラー・オプションおよびディレクティブについて説明します。

コマンド行でコンパイラー・オプションを指定することができます。また、アプリケーション・ソース・ファイルに埋め込まれたプラグマ・ディレクティブによりコンパイラーの動作を変更することもできます。 これらのコンパイラー・オプションの詳細説明および使用法に関する情報については、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。

表 10. 新規または変更されたコンパイラー・オプションおよびディレクティブ

オプションまたはディレクティブ	説明
-qstrict	多くのサブオプションが **-qstrict** オプションに追加されることで、最適化の制御、および厳密なプログラム・セマンティクスに違反するトランスフォーメーションの制御が向上しました。詳しくは、40 ページの『パフォーマンスおよび最適化』を参照してください。
-qshowmacros	**-E** オプションとともに使用した場合、**-qshowmacros** オプションはプリプロセスされた出力をマクロ定義で置き換えます。定義済みマクロおよびユーザー定義マクロの出力をさらに正確に制御するためサブオプションが提供されています。
-qreport	自動並列化またはベクトル化を使用可能にするコンパイラー・オプションと共に使用すると、 **-qreport** オプションはループに含まれるストリームの数を報告し、ループが、非ストライド・ワン参照によって SIMD ベクトル化されないときに情報を作成するようになりました。

表 10. 新規または変更されたコンパイラー・オプションおよびディレクティブ (続き)

オプションまたはディレクティブ	説明
-qsmp	**-qsmp=omp** が有効になっているとき、OpenMP API 3.0 の追加機能が使用できるようになりました。詳しくは、52 ページの『OpenMP 3.0』を参照してください。
-qtimestamps	このオプションは生成されたバイナリーからタイム・スタンプを除去するために使用されます。
-qtls	スレッド・ローカル・ストレージ・サポートが拡張されて、`__attribute__((tls-model("string")))` が組み込まれています。ここで、*string* は、 `local-exec`、`initial-exec`、`local-dynamic`、または `global-dynamic` のいずれかです。
-qinfo	サブオプション `als` および `noals` は、ANSI 別名割り当て規則の考えられる違反を報告する (または報告しない) ために **qinfo** オプションに追加されました。

事前定義マクロ

このセクションには、このリリースで追加された事前定義マクロについての情報が記載されています。

XL C/C++ V10.1 では、次の 4 つのマクロが追加されました。

_ILP32 __ILP32__
: コンパイルが `long int`、`int` およびポインターがすべて 32 ビットを使用するターゲット用であるときにのみ 1 に定義される。その他の場合は定義されない。

_LP64 __LP64__
: `long int` およびポインターの両方が 64 ビットを使用し、`int` が 32 ビットを使用するターゲット用のコンパイルである場合にのみ 1 に定義される。その他の場合は定義されない。

現在では、コンパイラーは `__C99_COMPLEX_HEADER__` マクロをサポートしません。

XL C/C++ の定義済みマクロの完全なリストは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー定義済みマクロ』 を参照してください。

第 4 章 XL C/C++ のセットアップとカスタマイズ

XL C/C++ の完全な前提条件およびインストール情報については、「*XL C/C++ インストール・ガイド*」の 『XL C/C++ のインストール前の作業』 を参照してください。

カスタム・コンパイラー構成ファイルの使用

デフォルトの構成ファイルを変更するか、ユーザー独自の構成ファイルを作成して、コンパイラーの設定およびオプションをカスタマイズできます。

以下のオプションを使用して、コンパイラー設定をカスタマイズできます。

- XL C/C++ コンパイラーのインストール・プロセスによって、デフォルトのコンパイラー構成ファイルが作成されます。この構成ファイルを直接変更して、特定のニーズのためのデフォルト・オプションを追加することができます。ただし、後でコンパイラーに更新を適用する場合、ユーザーが行ったすべての変更を、新規にインストールされた構成ファイルに対して再適用する必要があります。
- デフォルトの構成ファイルをオーバーライドするか補完する独自のカスタム構成ファイルを作成できます。コンパイラーは、デフォルト構成ファイル内で指定されているコンパイラー設定のほか、ユーザーのカスタム構成ファイル内で指定するコンパイラー設定を認識および解決できます。 後でデフォルト構成ファイルの設定に影響を与える可能性があるコンパイラーの更新は、カスタム構成ファイル内の設定に影響を与えません。

詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『カスタム・コンパイラー構成ファイルの使用』を参照してください。

コンパイラー使用状況のトラッキングおよびレポート作成の構成

コンパイラー構成ファイルのほかに、使用状況のトラッキングおよびレポート作成機能用に、別個の構成ファイルがあります。デフォルトで、使用状況のトラッキングは使用不可に設定されていますが、この構成ファイルの項目を変更することで使用可能に設定できます。このファイルを使用して、使用状況のトラッキングのその他のさまざまな側面を構成することもできます。

コンパイラー構成ファイルは使用状況のトラッキング構成ファイルとは別個のものですが、このファイルには、使用状況のトラッキング構成ファイルの場所を指定する項目が含まれているため、コンパイラーによってこのファイルを検出することができます。

使用状況のトラッキングおよびレポート作成機能の構成方法について詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラー使用状況のトラッキングおよびレポート作成』を参照してください。

第 5 章 XL C/C++ によるアプリケーションの開発

C/C++ アプリケーション開発は、編集、コンパイル、リンク、および実行というサイクルを繰り返すことで成り立っています。デフォルトでは、コンパイルとリンクは単一ステップに結合されています。

注:

1. コンパイラーを使用する前に、まず XL C/C++ が適切にインストールされ、構成されていることを確認する必要があります。詳しくは、「*XL C/C++ インストール・ガイド*」を参照してください。
2. C/C++ プログラムの作成について学習するには、「*XL C/C++ ランゲージ・リファレンス*」を参照してください。

コンパイラー・フェーズ

標準的なコンパイラー呼び出しは、以下に挙げる活動のいくつかまたはすべてを順序どおりに実行します。リンク時の最適化のために、一部の活動がコンパイル中に複数回実行されます。各コンパイル・コンポーネントが実行されるたびに、その結果がシーケンス内の次のステップに送られます。

1. ソース・ファイルのプリプロセッシング
2. コンパイル (指定されるコンパイラー・オプションによって異なりますが、例えば以下のフェーズで構成されます)
   a. フロントエンド構文解析およびセマンティック分析
   b. 高水準最適化
   c. 低水準最適化
   d. レジスター割り振り
   e. 最終アセンブリー
3. アセンブリー (**.s**) ファイル、および、プリプロセス後の未プリプロセス・アセンブラー (**.S**) ファイルのアセンブル
4. 実行可能アプリケーションを作成するためのオブジェクト・リンク

コンパイラーがこれらのフェーズをステップスルーするのを確認するには、ユーザーのアプリケーションをコンパイルするときに **-v** コンパイラー・オプションを指定します。 コンパイラーが各フェーズにかける時間を確認するには、 **-qphsinfo** を指定します。

C/C++ ソース・ファイルの編集

C/C++ ソース・プログラムを作成するには、 ご使用のシステムで使用可能な任意のテキスト・エディターを使用することができます。

ソース・プログラムは、認識されたファイル名接尾部を使用して保管する必要があります。 XL C/C++ によって認識されたサフィックスのリストについては、 62 ページの『XL C/C++ 入力および出力ファイル』を参照してください。

C または C++ ソース・プログラムが有効なプログラムであるためには、「*XL C/C++ ランゲージ・リファレンス*」で指定されている言語定義に準拠している必要があります。

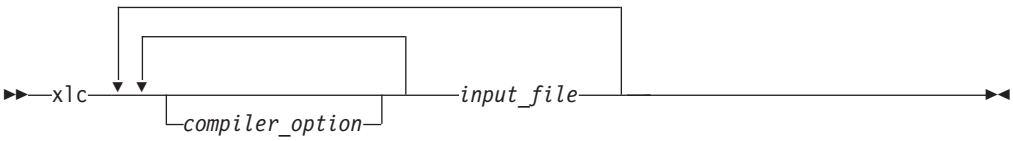
XL C/C++ によるコンパイル

XL C/C++ は、コマンド行コンパイラーです。呼び出しコマンドおよびオプションは、特定の C/C++ アプリケーションのニーズにしたがって選択できます。

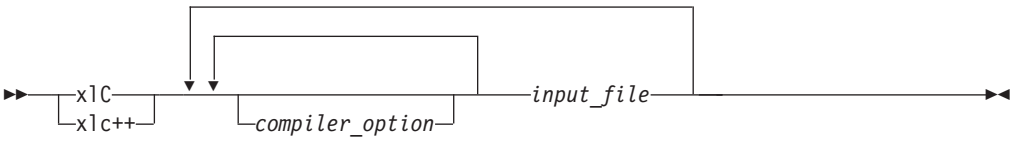
コンパイラーの呼び出し

コンパイラー呼び出しコマンドは、C または C++ ソース・ファイルをコンパイルして、任意の **.s** ファイルおよび **.S** ファイルをアセンブルし、オブジェクト・ファイルとライブラリーを 1 つの実行可能プログラムにリンクするのに必要なすべての手順を実行します。

C ソース・プログラムをコンパイルするには、次の基本呼び出し構文を使用します。



C++ ソース・プログラムをコンパイルするには、次の基本呼び出し構文を使用します。



大部分のアプリケーションでは、**xlc**、**xlc++**、、または対応するスレッド・セーフ呼び出しを使用してコンパイルします。**xlc++** を使用して C または C++ プログラム・ソースをコンパイルできますが、**xlc** で C++ ファイルをコンパイルするとリンクまたはランタイムのエラーになることがあります。これは、C++ コードに必要なライブラリーが、リンカーの C コンパイラーによる呼び出し時に指定されないためです。

特殊なコンパイルのニーズを満たし、主に C または C++ 言語のさまざまなレベルおよび拡張機能に対する明示的なコンパイル・サポートを提供するために、他にも呼び出しコマンドを使用できます。 GNU コンパイル環境から XL C/C++ にマイグレーションする開発者を支援するための特殊な呼び出しなどの、使用可能なコンパイラー呼び出しコマンドについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の 『コンパイラーの呼び出し』を参照してください。

並列化 XL C/C++ アプリケーションのコンパイル

XL C/C++ には、マルチプロセッサー環境で使用する並列化アプリケーションをコンパイルするためのスレッド・セーフなコンパイラー呼び出しコマンドがあります。

これらの呼び出しは、それらがコンパイルされたオブジェクトをスレッド・セーフ・コンポーネントおよびライブラリーにリンクしバインドすることを除いて、対応する基本コンパイラー呼び出しと同様です。汎用的なXL C/C++スレッド・セーフ・コンパイラー呼び出しは次のとおりです。

- xlC_r
- xlc++_r
- xlc_r

XL C/C++ は、特定のコンパイル要件を満たすための、追加スレッド・セーフ呼び出しを提供します。詳しくは、 「*XL C/C++ コンパイラー・リファレンス*」の『コンパイラーの呼び出し』を参照してください。

注: これらのコマンドのいずれかを単独で使用することは、並列処理を暗黙指定することを意味しません。コンパイラーが OpenMP ディレクティブを認識し並列処理を活動化するためには、**-qsmp** コンパイラー・オプションも指定する必要があります。つまり、これらのスレッド・セーフ呼び出しコマンドのうちの 1 つのみとともに **-qsmp** オプションを指定する必要があります。 **-qsmp** を指定すると、ドライバーは構成ファイルのアクティブ・スタンザにある smp ライブラリー行で指定されたライブラリーにリンクします。

並列処理アプリケーションについて詳しくは、 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『プログラムの並列処理』 を参照してください。

コンパイラー・オプションの指定

コンパイラー・オプションは、コンパイラー特性の設定、作成されるオブジェクト・コードの記述、出される診断メッセージの制御、および一部のプリプロセッサー関数の実行など、さまざまな機能を実行します。

コンパイラー・オプションは、次のいずれかの方法で、または複数の方法を組み合わせて指定できます。

- コマンド行コンパイラー・オプションを指定して、コマンド行で
- ソース・コードでディレクティブ・ステートメントを使用して
- Make ファイルで
- コンパイラー構成ファイルにあるスタンザで

コンパイラー・オプションをリンカー、アセンブラー、およびプリプロセッサーに渡すこともできます。

コンパイラー・オプションおよびその使用法について詳しくは、 「*XL C/C++ コンパイラー・リファレンス*」の 『コンパイラー・オプション・リファレンス』 を参照してください。

コンパイラー・オプションの優先順位

複数のコンパイラー・オプションを指定した場合、オプションの競合および非互換が起きる可能性があります。このような競合を整合性のある方法で解決するために、通常、コンパイラーは次の一般的な優先順位をほとんどのオプションに適用します。

1. ソース・ファイルのディレクティブ・ステートメントは、コマンド行設定値をオーバーライド する。
2. コマンド行コンパイラー・オプション設定値は、構成ファイル設定値をオーバーライド する。
3. 構成ファイル設定値は、デフォルト設定値をオーバーライド する。

一般に、コンパイラーを呼び出すときにコマンド行上で同じコンパイラー・オプションが複数回指定されると、最後に指定されたオプションが優先されます。

注: コンパイラー・オプションの中には、 **-I** オプションのように、上記の優先順位に従わないものもあります。コンパイラーは、コマンド行で **-I** を使用して指定されたディレクトリーを検索する前に、xlc.cfg ファイル内の **-I** で指定された任意のディレクトリーを検索します。 **-I** オプションはプリエンプティブではなく累積型です。累積的動作を行う他のオプションは、**-R** および **-l** (小文字の L) です。

gxlc および gxlc++ による GNU C/C++ コンパイラー・オプションの再使用

XL C/C++ には、gxlc および gxlc++ コマンドなどを含め、GNU C/C++ コンパイラーから XL C/C++ への移行を支援するさまざまな機能が組み込まれています。

各 gxlc および gxlc++ ユーティリティーは、GNU C または C++ コンパイラー・オプションを受け取り、それを同等の XL C/C++ オプションに変換します。両方のユーティリティーは、XL C/C++ オプションを使用して **xlc** または **xlc++** の呼び出しコマンドを作成し、次にそれがコンパイラーを呼び出すのに使用されます。これらのユーティリティーは、ユーザーが、以前に GNU C/C++ を使用して開発されたアプリケーション用に作成された Make ファイルを再使用するのを援助するために提供されています。 ただし、 XL C/C++ の機能を十分活用するために、XL C/C++ の呼び出しコマンドおよびその関連オプションを使用できます。

gxlc および gxlc++ のアクションは `gxlc.cfg` 構成ファイルによって制御されます。XL C/C++ に対応するもののある GNU C/C++ オプションは、このファイルに示されています。 すべての GNU オプションが対応する XL C/C++ オプションをもっているわけではありません。 gxlc および gxlc++ は、変換されなかった入力オプションについて警告を戻します。

gxlc および gxlc++ オプション・マッピングは変更可能です。 gxlc または gxlc++ 構成ファイルの使用については、「*XL C/C++ コンパイラー・リファレンス*」内の『gxlc および gxlc++ による GNU C/C++ コンパイラー・オプションの再使用』を参照してください。

XL C/C++ 入力および出力ファイル

これらのファイル・タイプが XL C/C++ によって認識されます。

コンパイラーが使用する以下のファイル・タイプおよび追加のファイル・タイプについて詳しくは、 「*XL C/C++ コンパイラー・リファレンス*」の『入力ファイルのタイプ』および「*XL C/C++ コンパイラー・リファレンス*」の『出力ファイルのタイプ』を参照してください。

表 11. 入力ファイル・タイプ

ファイル名拡張子	説明
.c	C ソース・ファイル
.C、.cc、.cp、 .cpp、.cxx、.c++	C++ ソース・ファイル
.i	プリプロセスされたソース・ファイル
.o	オブジェクト・ファイル
.s	アセンブラー・ファイル
.S	プリプロセスされていないアセンブラー・ファイル
.so	共有オブジェクトまたはライブラリー・ファイル

表 12. 出力ファイル・タイプ

ファイル名拡張子	説明
a.out	コンパイラーによって作成される実行可能ファイルのデフォルト名
.d	Make 依存関係ファイル
.i	プリプロセスされたソース・ファイル
.lst	リスト・ファイル
.o	オブジェクト・ファイル
.s	アセンブラー・ファイル
.so	共有オブジェクトまたはライブラリー・ファイル

XL C/C++ によるコンパイル済みアプリケーションのリンク

デフォルトでは、ユーザーは、XL C/C++ プログラムをリンクするのに、特別なことは何もする必要はありません。コンパイラー呼び出しコマンドは、自動でリンカーを呼び出して実行可能出力ファイルを生成します。

例えば、次のコマンドを使用して `file1.C` および `file3.C` をコンパイルし、オブジェクト・ファイル `file1.o` および `file3.o` を作成できます。 `file2.o` を含むすべてのオブジェクト・ファイルの処理がリンカーに依頼されて 1 つの実行可能ファイルが作成されます。

```
xlc++ file1.C file2.o file3.C
```

別個のステップでのコンパイルおよびリンク

後でリンクできるオブジェクト・ファイルを作成するには、**-c** オプションを使用します。

```
xlc++ -c file1.C              # Produce one object file (file1.o)
xlc++ -c file2.C file3.C      # Or multiple object files (file1.o, file3.o)
xlc++ file1.o file2.o file3.o  # Link object files with default libraries
```

プログラムのコンパイルおよびリンクについて詳しくは、以下を参照してください。

- 「*XL C/C++ コンパイラー・リファレンス*」の 「リンク」
- 「*XL C/C++ 最適化およびプログラミング・ガイド*」の『ライブラリーの構成』

動的および静的リンク

XL C/C++ を使用すれば、ご使用のプログラムは、動的リンクと静的リンクの両方のためのオペレーティング・システム機能の利点を利用できるようになります。

動的リンクとは、プログラムが初めて実行されるときに、なんらかの外部ルーチン用のコードが探し出されてロードされることを意味します。共用ライブラリーを使用するプログラムをコンパイルすると、デフォルトではプログラムに動的にリンクされます。動的にリンクされたプログラムでは、共用ライブラリーのルーチンを複数のプログラムが使用していても、ディスク・スペースも仮想メモリーも少なくて済みます。リンクが行われているときに、それらのプログラムについては、ライブラリー・ルーチンとの命名上の競合を回避するためになんらかの特別な予防措置を講じる必要はありません。いくつかのプログラムが同時に同じ共有ルーチンを使用する場合は、静的にリンクされたプログラムよりも良好に動作するよう設計されています。動的リンクを使用することで、共用ライブラリー内のルーチンを再リンクせずにアップグレードできます。

このリンク形式はデフォルトなので、これをオンにするのに追加のオプションは必要ありません。

静的リンクとは、プログラムによって呼び出されるすべてのルーチン用のコードが実行可能ファイルの一部になることを意味します。

静的にリンクされたプログラムは移動して XL C/C++ ランタイム・ライブラリーがないシステム上で実行することができます。静的にリンクされたプログラムがライブラリー・ルーチンを何回も呼び出したり、多数の小さなルーチンを呼び出したりする場合、それらのプログラムは動的にリンクされたプログラムよりも良好に動作する可能性があります。ライブラリー・ルーチンとの命名の競合を回避したい場合は、プログラム内のデータ・オブジェクトおよびルーチンの名前を選択するときに、何らかの予防措置をとる必要があります。また、それらのプログラムをあるレベルのオペレーティング・システムでコンパイルして、別のレベルのオペレーティング・システムで実行すると、機能しない可能性があります。

コンパイル済みアプリケーションの実行

プログラムがコンパイルおよびリンクされた後で、生成された実行可能ファイルをコマンド行で実行できます。

XL C/C++ コンパイラーによって作成されたプログラム実行可能ファイルのデフォルト・ファイル名は、**a.out** です。 **-o** コンパイラー・オプションを指定して別の名前を選択することができます。

間違ったコマンドを誤って実行することがあり得るので、プログラム実行可能ファイルに、システムまたはシェルのコマンドと同じ名前 (例えば `test` または `cp` など) を付けないでください。プログラム実行可能ファイルにシステム・コマンドま

たはシェル・コマンドと同じ名前を付けるように決めた場合には、実行可能ファイルが存在するディレクトリーに、./test といったパス名を指定することによって、そのプログラムを実行することができます。

プログラムを実行するためには、コマンド行にプログラム実行可能ファイルの名前を任意の実行時引数と一緒に入力します。

実行の取り消し

プログラムの実行を中断するには、プログラムがフォアグラウンドにある間に **Ctrl+Z** キーを押してください。実行を再開するには、**fg** コマンドを使用してください。

実行中のプログラムを取り消すには、プログラムがフォアグラウンドにある間に **Ctrl+C** キーを押してください。

実行時オプションの設定

環境変数の設定値を使用して、XL C/C++ コンパイラーで作成されたアプリケーションの特定の実行時オプションおよび動作を制御することができます。 一部の環境変数は、実際の実行時の動作を制御しなくても、アプリケーションの動作に影響を与えることがあります。。

環境変数の詳細、および環境変数が実行時にアプリケーションにどのような影響を与えるかに関する詳細については、「*XL C/C++ インストール・ガイド*」を参照してください。

他のシステムでのコンパイル済みアプリケーションの実行

XL C/C++ コンパイラーを使用して開発されたアプリケーションを、そのコンパイラーがインストールされていない他のシステムで実行する場合は、そのシステムにランタイム環境をインストールするか、アプリケーションを静的にリンクする必要があります。

最新の XL C/C++ ランタイム環境 PTF イメージは、ライセンス情報および使用情報とともに、「XL C/C++ for Linux サポート・ページ」から取得できます。

XL C/C++ コンパイラー診断エイド

XL C/C++ は、ユーザーのアプリケーションをコンパイルしているときに問題に遭遇すると、診断メッセージを出します。ユーザーは、そのような問題を識別して訂正する援助として、コンパイラー出力リストで提供されるこれらのメッセージおよびその他の情報を使用することができます。

アプリケーションの問題の解決に役立つリスト作成、診断、関連コンパイラー・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の以下のトピックを参照してください。

- コンパイラー・メッセージおよびリスト表示
- エラー検査およびデバッグ・オプション
- リスト、メッセージ、およびコンパイラー情報のオプション

コンパイル済みアプリケーションのデバッグ

シンボリック・デバッガーを使用して、 XL C/C++ を使用してコンパイルされたアプリケーションをデバッグすることができます。

コンパイル時に、**-g** オプションまたは **-qlinedebug** オプションを指定して、コンパイルした出力にデバッグ情報を組み込むように XL C/C++ コンパイラーに指示できます。**-g** の場合、さまざまなレベルを使用して、デバッグ機能とコンパイラー最適化の間のバランスを取ることもできます。デバッグ・オプションについて詳しくは、「*XL C/C++ コンパイラー・リファレンス*」の『エラー検査およびデバッグ』を参照してください。

次に、 **gdb** またはその他の任意のシンボリック・デバッガーを使用して、コンパイル済みアプリケーションの動作をステップスルーし、検査することができます。

最適化されたアプリケーションをデバッグすると、特殊な問題が発生します。 高度に最適化されたアプリケーションをデバッグするときは、**-qoptdebug** コンパイラー・オプションの使用を検討してください。 コードの最適化について詳しくは、「*XL C/C++ 最適化およびプログラミング・ガイド*」の 『アプリケーションの最適化』 を参照してください。

使用されている XL C/C++ のレベルの判別

使用している XL C/C++ のバージョンおよびリリース・レベルを表示するには、**-qversion** コンパイラー・オプションを指定してコンパイラーを呼び出します。

例えば、詳細なバージョン情報を表示するには、次の コマンド を入力します。

```
xlc++-qversion=verbose
```

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario L6G 1C7
Canada

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標または登録商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴは、 Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

C

G

M

O

S

X

［特殊文字］

プログラム番号: 5765-J08; 5725-C73

Printed in Japan

SA88-5392-00

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21