IBM XL C/C++ for Linux, V11.1

はじめに

バージョン 11.1

GI88-4231-00
(英文原典：GI11-7913-00)

IBM XL C/C++ for Linux, V11.1

はじめに

バージョン 11.1

GI88-4231-00
(英文原典：GI11-7913-00)

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、 49 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, V11.1 (プログラム 5724-X14)、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： GI11-7913-00
IBM XL C/C++ for Linux, V11.1
Getting Started with XL C/C++
Version 11.1

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

第1版第1刷 2010.5

目次

本書について

本書には、IBM® XL C/C++ for Linux®, V11.1 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。

本書の全体を通じて、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: C または C++ プログラミング言語の構文、セマンティクス、および IBM 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

書体の規則

次の表は、IBM XL C/C++ for Linux, V11.1 の情報で使用されている書体の規則を説明しています。

表 1. 書体の規則

書体	示す内容	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	コンパイラーは、基本的な呼び出しコマンド、 **xlc** および **xlC** (**xlc++**) を提供しています。また、他のコンパイラー呼び出しコマンドもいくつか提供しており、さまざまな C/C++ 言語水準およびコンパイル環境をサポートしています。
イタリック	対応する実際の名前または値は、ユーザーが提供することになるパラメーターまたは変数。イタリックは、また、新規の用語を導入する場合にも使用されます。	要求された *size* よりも大きいものを戻す場合には、*size* パラメーターを更新すること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、コンパイラー組み込み関数、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名。	myprogram.c をコンパイルし、最適化するには、 `xlc myprogram.c -O3` と入力します。

エレメントの修飾 (アイコン)

本書で説明されている大部分の機能は、C 言語と C++ 言語の両方に適用されます。ある機能が一方の言語専用の機能である場合、または言語間で機能の働きが異なる場合には、言語要素の表現は、テキストのセグメントを描写するためアイコンを次のように使用します。

表 2. エレメントの修飾

限定子/アイコン	意味
C のみ、または C のみ開始   C のみ終了	テキストは、C 言語のみでサポートされる機能を説明するか、または C 言語固有の動作を説明します。
C++ のみ、または C++ のみ開始   C++ のみ終了	テキストは、C++ 言語のみでサポートされる機能を説明するか、または C++ 言語固有の動作を説明します。

表 2. エレメントの修飾 (続き)

限定子/アイコン	意味
IBM 拡張開始   IBM 拡張終了	テキストは、標準の言語仕様に対する IBM 拡張である機能を説明します。
C++0x、または C++0x 開始   C++0x 終了	テキストは、C++0x の一部として標準 C++ に導入された機能を説明します。

構文図

本書全体にわたって、構文図が XL C/C++ 構文を図示しています。このセクションは、構文図の解釈と使用に役立ちます。

- 構文図は、線の経路に従って、左から右へ、上から下へ読みます。

  ▸▸── 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始まりを示します。

  ──▸ 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構文が次の行に続くことを示します。

  ▸── 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わりを示します。

  完全なコマンド、ディレクティブ、または文以外の構文単位の断片図は、|── 記号で始まり、──| 記号で終わります。
- 必須項目は、水平線上 (主経路上) に示されます。

```
▸▸─keyword─required_argument─────────────────────────────────────▸◂
```

- オプショナル項目は、主経路の下に示されます。

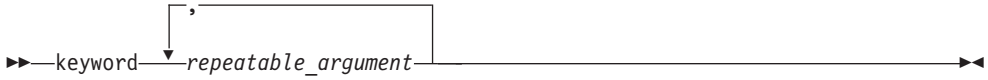
```
▸▸─keyword──┬──────────────────┬──────────────────────────────────▸◂
            └─optional_argument─┘
```

- 複数の項目の中から選択できる場合には、それらの項目は縦に積み重ねて (スタックの形で) 示されます。

  複数の項目から 1 つを*選択しなければならない* 場合は、縦の並びの中のいずれか 1 つの項目がメインパス上に表示されます。

複数の項目の中から 1 つを選択することがオプショナルである場合には、スタック全体が主経路の下に示されます。

- 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)。

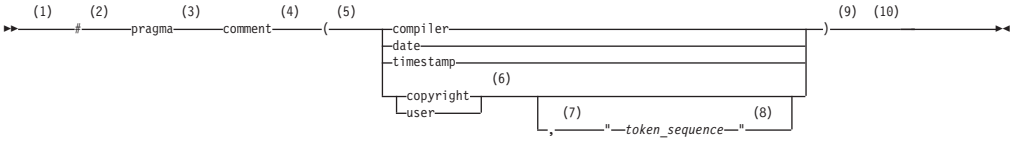


- デフォルトである項目は、主経路の上に示されます。

- 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、またはその他のそのような記号が示されている場合には、それら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ています。



注:

1 これが構文図の始まりです。

2 記号 # が先頭になければなりません。

3 キーワード pragma が # 記号の後に続く必要があります。

4 プラグマ comment の名前がキーワード pragma の後に続く必要があります。

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、次に示すタイプの 1 つのみとして入力する必要があります。compiler、date、timestamp、copyright、または user。

7 コメント・タイプ `copyright` または `user` とオプションの文字ストリングの間には、コンマが必要です。

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、上記の図と構文的に一致しており、正しく入力された例です。

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

本書での例

本書に記載されている例は、特に注記がない限り、ストレージを節約すること、エラーの有無を検査すること、速いパフォーマンスを達成すること、また特定の結果を得るために可能なあらゆる方法を明示することのいずれも試すことがない、単純な形式でコーディングされています。

本書での例は、*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のインストール中に実行される手順をほとんどそのまま、または変更しないで示すことを意図したものです。

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、製品情報を、以下の形式で提供します。

- 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++ ディレクトリーにあり、またインストール 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 man ページ

  man ページは、製品に提供されているコンパイラー呼び出しおよびすべてのコマンド行ユーティリティーのために提供されています。 man ページをインストールし、それにアクセスするための指示は、「*IBM XL C/C++ for Linux, V11.1 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

  検索可能な HTML ファイルのインフォメーション・センターは、ネットワーク上で起動することができ、リモート側で、またはローカルでアクセスできます。オンライン・インフォメーション・センターをインストールし、それにアクセスするための指示は、「*IBM XL C/C++ for Linux, V11.1 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・センターは、Web 上の http://publib.boulder.ibm.com/infocenter/lnxpcomp/v111v131/index.jspで表示可能です。

- PDF 文書

  PDF 文書は、デフォルトでは /opt/ibmcmp/vacpp/11.1/doc/*LANG*/pdf/ ディレクトリー (ここで、*LANG* は en_US、zh_CN、 または ja_JP のいずれか) にあります。PDF ファイルは、以下の Web サイトhttp://www.ibm.com/software/awdtools/xlcpp/linux/library/でも入手できます。

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

表 3. XL C/C++ PDF ファイル

資料の表題	PDF ファイル名	説明
IBM XL C/C++ for Linux, V11.1 インストール・ガイド, GI88-4232-00	install.pdf	XL C/C++ をインストールし、ユーザーの環境を基本的なコンパイルおよびプログラム実行用に構成するための情報が記載されています。
IBM XL C/C++ for Linux, V11.1 はじめに, GI88-4231-00	getstart.pdf	XL C/C++ 製品の概要説明のほか、ユーザー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ます。
IBM XL C/C++ for Linux, V11.1 コンパイラー・リファレンス, SC88-8382-00	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、および並列処理に使用されるものを含む、組み込み関数についての情報が含まれます。
IBM XL C/C++ for Linux, V11.1 ランゲージ・リファレンス, SC88-8383-00	langref.pdf	ポータビリティーおよび非独占標準への準拠に対応した言語拡張機能などの、IBM がサポートする C および C++ プログラミング言語についての情報の情報が含まれます。
IBM XL C/C++ for Linux, V11.1 最適化およびプログラミング・ガイド, SC88-8385-00	proguide.pdf	拡張プログラミングのトピック (アプリケーション・ポーティング、 FORTRAN コードでの言語間呼び出し、ライブラリー開発、アプリケーション最適化および並列化など)、および XL C/C++ ハイパフォーマンス・ライブラリーの情報が含まれます。

  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、(ライセンス条項に従うことにより) Adobe Web サイト (http://www.adobe.com) からダウンロードできます。

IBM Redbooks® の資料、ホワイト・ペーパー、チュートリアル、およびその他の記事を含め、XL C/C++ に関連する情報が、次の Web サイトで使用可能です。

http://www.ibm.com/software/awdtools/xlcpp/linux/library/

パフォーマンス、生産性、および移植性の向上について詳しくは、C/C++ café (http://www-949.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 14882:1998*、別名 *C++98*。
- *Information Technology - Programming languages - C++, ISO/IEC 14882:2003(E)*、別名 *標準 C++*。
- *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・テクニカル・レポートは C 標準委員会で承認され、サイト http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1040.pdf で使用可能です。
- *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。
- *Military Standard Fortran DOD Supplement to ANSI X3.9-1978, MIL-STD-1753*(米国、国防総省標準)。XL Fortran は、Fortran 90 標準に順次取り込まれてもいるこの標準において文書化されている拡張のみをサポートしていることに注意してください。
- http://www.openmp.org で使用可能な「*OpenMP Application Program Interface Version 3.0*」。

その他の IBM 情報

- *ESSL for AIX V4.4 - ESSL for Linux on POWER V4.4 Guide and Reference*は、Engineering and Scientific Subroutine Library (ESSL) and Parallel ESSL の Web ページで入手できます。

その他の情報

- http://gcc.gnu.org/onlinedocs で使用可能な「*Using the GNU Compiler Collection*」

技術サポート

追加の技術サポートを http://www.ibm.com/software/awdtools/xlcpp/linux/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。

必要なものが見つからない場合には、compinfo@ca.ibm.com に E メールを出して問い合わせることができます (英文でのみ対応)。

XL C/C++ に関する最新の情報に関しては、 http://www.ibm.com/software/awdtools/xlcpp/linux/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, V11.1 は高機能で高性能なコンパイラーであり、 C および Fortran プログラムでの言語間呼び出しを含め、複雑で計算負荷の高いプログラムの開発に使用することができます。

この章では、XL C/C++ のコンパイラーの機能についての概要を述べます。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, V11.1 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®/L™、IBM Blue Gene®/P™、 Cell Broadband Engine™ アーキテクチャー、IBMi、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。

ハードウェアおよびオペレーティング・システムのサポート

IBM XL C/C++ for Linux, V11.1 は、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。

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 POWER4™、POWER5™、POWER5+™、POWER6®、POWER7™、PowerPC®、および PowerPC 970 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、コンパイラーは、コンパイルしたアプリケーションの実行に使用されるハードウェアのタイプに特化した、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を備えています。

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ときに使用できる、さまざまなコマンドが用意されています。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、ほとんどの呼び出しコマンドについて、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。さまざまなカテゴリーのオプションは、ユーザーがアプリケーションをデバッグし、アプリケーションのパフォーマンスを最適化して調整し、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のために言語レベルおよび言語拡張を選択し、それなしではソース・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行うのに役立ちます。

XL C/C++ では、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指示ステートメントの組み合わせを通じてコンパイラー・オプションを指定します。

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。

ユーザーのコンパイル上の必要によって、XL C/C++ が提供するデフォルトの設定値以外のコンパイラー・オプション設定値を指定することが要求されるという状況がしばしば起こり得ます。その場合は Make ファイルを使用して、コンパイラー・オプションを定義することができます。または、代わりにカスタム構成ファイルを作成して、頻繁に使用するコンパイラー・オプション設定の独自のセットを定義することができます。

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、39 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

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。本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、使用状況のトラッキングおよびレポート作成機能の機能性を集中的に管理できます。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。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

コンパイラーは、C/C++ に関する以下のプログラム言語仕様をサポートします。

- ISO/IEC 9899:1999 (C99)
- ISO/IEC 9899:1990 (C89 と呼ばれる)
- 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)

標準化された言語レベルに加えて、XL C/C++ は、以下を含む言語拡張もサポートします。

- 並列化されたプログラミングをサポートする OpenMP V3.0
- 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++ 言語拡張のサブセット
- C++0x

  詳しくは、 「*XL C/C++ for Linux はじめに*」の C++0xを参照してください。

C/C++ 言語仕様および言語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『言語レベルと言語拡張』を参照してください。

GNU との互換性

XL C/C++ は、**gcc**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++** 呼び出しコマンドが 選択された GNU コンパイラー・オプションと一緒に使用される場合に使用可能です。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。

これらの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。ユーザ

ーは、この構成ファイルをカスタマイズして、ユーザーのあらゆる固有のコンパイル要件をよりいっそう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++ は、コンパイラーの呼び出しコマンドを提供することで、ユーザーの既存の C/C++ ソース・コードに対する投資を保護するのに役立ちます。この呼び出しコマンドは、特定の言語レベルに対してユーザーのアプリケーション・コードをコンパイルし、

ユーザーは、また、**-qlanglvl** コンパイラー・オプションを使用して特定の言語レベルを指定することができ、コンパイラーは、ユーザーのソース・プログラム内の言語または言語拡張エレメントがその言語レベルに準拠しない場合、警告、エラー、および重大エラー・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、サポートされているプロセッサー・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。 ユーザーは、MASS ライブラリーを選択して、広範囲のプロセッサー上での高性能コンピューティングをサポートするか、または特定のプロセッサー・ファミリーをサポートするために調整されたライブラリーを選択することができます。

MASS ライブラリー機能は、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、スレッド・セーフであり、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・ルーチンよりも改良されています。 これらのライブラリーは、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に、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すると、以下のことが行えます。

- 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

以下も、 XL C/C++ と一緒に配送されます。

- 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

XL C/C++ for Linux V11.1 は、Boost V1.40.0 ライブラリーを部分的にサポートしています。また、Boost 1.40.0 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・ファイルにアクセスするには、そのページ (http://www.ibm.com/support/docview.wss?uid=swg27006911) に進み、必要な Boost 変更ファイルをダウンロードするセクションのリンクをたどります。

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/software/awdtools/xlcpp/linux/support/) で検索してください。

ツールおよびユーティリティー

XL C/C++ に組み込まれている、多くのツールおよびユーティリティーがあります。

new_install
IBM XL C/C++ for Linux, V11.1 をインストールした後、このユーティリティーを実行すると、ご使用のシステムでコンパイラーが使用できるように構成されます。

vac_configure
この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するために使用します。

cleanpdf コマンド
Profile-Directed Feedback (PDF) に関連したコマンド。 **cleanpdf** は、PDF データが書き込まれるディレクトリーからすべてのプロファイル情報を除去します。

mergepdf コマンド
Profile-Directed Feedback (PDF) に関連したコマンド。 **mergepdf** 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
cleanpdf コマンドの現在の動作は、**resetpdf** 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存されています。

showpdf コマンド
showpdf コマンドは、Profile-Directed Feedback トレーニング実行 (オプション **-qpdf1** および **-qshowpdf** のもとでのコンパイル) で実行されるすべてのプロシージャーの呼び出しおよびブロック数を表示します。

gxlc および gxlc++ ユーティリティー
gxlc および **gxlc++** 呼び出しは、GNU C または GNU C++ 呼び出しコマンドを対応する **xlc** または **xlc++** コマンドに変換してから IBM XL C/C++ for Linux, V11.1 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、IBM XL C/C++ for Linux, V11.1 への変換を容易にすることにあります。

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** コマンドを使用すると、レポートの生成方法を制御できます。このツー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。

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 Architecture®の最適な使用のための、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・モードおよび 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。

XL C/C++ for Linux V10.1から、XL C/C++ は OpenMP API バージョン 3.0 仕様をサポートします。この機能によって提供されるサポートの概要については、29 ページの『OpenMP 3.0』 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』
- www.openmp.org

診断リスト

コンパイラー出力リストおよび XML 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ることがあります。

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、コンパイラーが行うことができたいくつかの最適化に関する情報や、いずれの最適化機会を逃したかに関する情報も、コンパイラーから XML 1.0 形式で取得できます。この情報を使用して、アプリケーションのチューニング、特に高性能アプリケーションのチューニングを行うときに、プログラミングの労力を減らすことができます。 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

コンパイル済みオブジェクトにデバッグ情報を含めるように XL C/C++ に指示することができます。デバッグ情報は、IBM Debugger for AIX、 **gdb**、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, V11.1 の新機能

このセクションでは、IBM XL C/C++ for Linux, V11.1 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。

オペレーティング・システムのサポート

このセクションには、サポートされるオペレーティング・システムに関する情報が記載されています。

IBM XL C/C++ for Linux, V11.1 は、IBM Power Systems™ サーバーでサポートされる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 1 (SLES 11 SP1)
- Red Hat Enterprise Linux 5.5 (RHEL 5.5)
- SUSE Linux Enterprise Server 10 Service Pack 2 (SLES 10 SP2)

POWER7 プロセッサーのサポート

IBM 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、『ベクトル型』 (「*XL C/C++ ランゲージ・リファレンス*」内)、および『ベクトル組み込み関数』 (「*XL C/C++ コンパイラー・リファレンス*」内) 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、 24 ページの『本リリースにおける新規組み込み関数』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++0x

C++0x は、新しい C++ プログラミング言語標準の作業ドラフトです。このリリースの XL C/C++ では、追加の C++0x 機能がサポートされています。

注: C++0x は、新しいバージョンの C++ プログラミング言語標準です。これはドラフトの標準であり、まだ全体が正式に採用されたものではありません。C++0x の実装は、ドラフトの C++0x 標準の IBM による解釈に基づいており、随時、予告なしに変更される場合があります。IBM では、以前のリリースとの互換性を維持するための試みは、特に行っていません。したがって、C++0x 言語拡張を継続的なプログラミング・インターフェースとしては利用しないでください。

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++0x 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・ネーム・スペース定義
- 静的アサーション
- 可変数引数テンプレート

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。

個別のサブオプション **-qlanglvl=autotypededuction** またはグループ・オプション **-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 『auto 型指定子 (C++0x)』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。

個別のサブオプション **-qlanglvl=c99longlong** またはグループ・オプション **-qlanglvl=extended0x** 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++0x で採用されている C99 プリプロセッサー機能

C++0x 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。

個別のサブオプション **-qlanglvl=c99preprocessor** またはグループ・オプション **-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『C++0x に採用された C99 プリプロセッサー機能 (C++0x)』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。

個別のサブオプション **-qlanglvl=decltype** またはグループ・オプション **-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『decltype(式) 型指定子 (C++0x)』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。

個別のサブオプション **-qlanglvl=delegatingctors** またはグループ・オプション **-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『委任コンストラクター (C++0x)』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** および **-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++0x 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。

個別のサブオプション **-qlanglvl=extendedfriend** またはグループ・オプション **-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『フレンド (C++ のみ)』を参照してください。

インライン・ネーム・スペース定義

インライン・ネーム・スペース定義は、初期 inlineキーワードを使用するネーム・スペース定義です。インライン・ネーム・スペースのメンバーを、そのネーム・スペースを含むエンクロージング・ネーム・スペース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。

個別のサブオプション **-qlanglvl=inlinenamespace** またはグループ・オプション **-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『インライン・ネーム・スペース定義 (C++0x)』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。

個別のサブオプション **-qlanglvl=static_assert** またはグループ・オプション **-qlanglvl=extended0x** を使用して、静的アサーション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++0x)』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。

個別のサブオプション **-qlanglvl=variadic[templates]** またはグループ・オプション **-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『可変数引数テンプレート (C++0x)』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** を使用してプログラムをコンパイルします。プログラムは標準データ・セットを使用して実行され、プロファイル作成データを生成します。次に、**-qpdf2** を使用してプログラムをもう一度コンパイルします。これによって、プログラムはプロファイル作成データに基づいて最適化されます。

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。 IBM XL C/C++ for Linux, V11.1からは、ソース・ファイルを変更した後にプロファイル作成データを使用できます。これを行うには、PDF プロセスの 2 番目のステージで、不整合のプロファイル作成データを使用してアプリケーションをコンパイルします。

3 つ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。これらの新規サブオプションを使用すると、パフォーマンスの向上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、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、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。

3 つの新規 **-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、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、および拡張された値のプロファイルをサポートします。 `-qpdf1=level=0|1|2` を使用してアプリケーションをコンパイルし、さまざまなレベルの最適化を使用して、プロファイル作成データを生成できます。

  注: `-qpdf1=level=0` と `-qpdf1=level=1` は、両方ともシングルパス・プロファイルをサポートしますが、`-qpdf1=level=2` はマルチパス・プロファイルをサポートします。

exename
: **-o** パラメーターを使用して指定する PDF ファイルの名前を生成します。

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、18 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、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 4. 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`。 **-qprefetch** は、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します。 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

リスト・レポートにセクションが追加され、プログラムを理解するのに役立てることができるようになりました。 **-qreport** オプションを **-qpdf2** オプションを指定して使用すると、以下の追加セクションが、`PDF Report` というタイトルのセクションにあるリスト・ファイルに追加されます。

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: このレポートを取得するには、**-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、最適化レベル **-qipa=level=2** または **-O5** を、**-qreport** とともに指定します。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 5. 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、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。

コンパイラー・オプションは、コマンド行上で、あるいはアプリケーションのソース・ファイルに組み込まれているディレクティブを通じて、指定することができます。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 6. 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/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。
-qipa	**-r -qipa=relink** を指定することで IPA 情報を保存しながら、再リンク可能なオブジェクトを生成できます。

表 6. 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/ディレクティブ	説明
-qlanglvl	C++0x 新しいサブオプションが **-qlanglvl** に追加されました。 • **-qlanglvl=autotypededuction**: 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++0x 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・ネーム・スペース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・ネーム・スペースのメンバーを、エンクロージング・ネーム・スペース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。

表 6. 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/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、データ再編成、および Profile-Directed Feedback 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 7. 非推奨のディレクティブおよびオプション

オプション/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。

表 7. 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/ディレクティブ	説明
-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。
-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。

本リリースにおける新規組み込み関数

このセクションでは、本リリースにおける新規の組み込み関数をリストします。

XL C/C++ によって提供され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

第 3 章 以前のバージョンで追加された機能拡張

このセクションでは、以前のリリースでコンパイラーに追加された機能拡張のリストを示します。

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。

オペレーティング・システムのサポート

IBM XL C/C++ for Linux, V10.1 は、IBM Power Systems サーバーでサポートされるオペレーティング・システムの Red Hat Enterprise Linux 5.2 (RHEL 5.2) および SUSE Linux Enterprise Server 10 SP 2 (SLES10 SP2) をサポートします。

事前定義マクロ

新規のマクロが 4 つあります。

_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。

C++0x

このリリースから、C++ プログラミング言語、具体的には、C++0x 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。この標準はまだ正式に公認されていませんが、いくつかの機能はサポートし始められています。

注: C++0x は、新しいバージョンの C++ プログラミング言語標準です。これはドラフトの標準であり、まだ全体が正式に採用されたものではありません。C++0x の実装は、ドラフトの C++0x 標準の IBM による解釈に基づいており、随時、予告なしに変更される場合があります。IBM では、以前のリリースとの互換性を維持するための試みは、特に行っていません。したがって、C++0x 言語拡張を継続的なプログラミング・インターフェースとしては利用しないでください。

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++0x の機能です。

C++ での C99 long long

本リリースの XL C/C++ for Linux V11.1 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。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の『整数および浮動小数点数のプロモーション』を参照してください。

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『言語レベルと言語拡張』を参照してください。

その他の XL C/C++ 言語関連の更新

Vector データ型

Vector 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

一部のフィーチャーおよび機能拡張を使用して、アプリケーションのパフォーマンス・チューニングと最適化に役立てることができます。

-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 8. 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、29 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。

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

コンパイラー・オプションは、コマンド行上で、あるいはアプリケーションのソース・ファイルに組み込まれているディレクティブを通じて、指定することができます。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 9. 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/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、16 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。
-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、29 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。

バージョン 9.0 に追加された機能拡張

このセクションでは、バージョン 9.0 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/C++ 言語関連の更新

C コンパイルのデフォルト言語レベルが変更され、 `long long` データ型で算術変換を行う場合の新規の動作が導入されました。

C におけるデフォルト言語レベルの変更 - extc99

デフォルトの **-qlanglvl** コンパイラー・オプションの設定は、**xlc** 呼び出しを使用して C コンパイラーを呼び出す場合は、**extc99** のままです。この変更によって、**extc99** サブオプションを明示的に指定せずに C99 機能およびヘッダーを使用することができます。

新規デフォルトの **-qlanglvl=extc99** 設定でコンパイルした場合、以下の問題が起こる場合があります。

- C99 では、ポインターは、restrict で修飾できます。そのため、restrict は識別子として使用できません。
- long long データの C99 における扱いは、 C89 における long long データの処理方法とは異なります。
- C99 ヘッダー・ファイルは、新しいマクロ LLONG_MAX を limits.h で、va_copy を stdarg.h で定義します。
- マクロ __STDC_VERSION__ の値が、199409 から 19990 に変更されています。

以前の **xlc** の動作に戻すためには、コンパイラーの呼び出し時に、**-qlanglvl=extc89** を指定します。

long long データ型での算術変換

XL C/C++ バージョン 9.0 では、long long 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。

次の場合には、算術式を想定します。

- 一方のオペランドの型が long long int または long long で、
- 他方のオペランドの型が unsigned long int であるが、その値は long long int または long long で表現できない。

以前のリリースの XL C/C++ では、両方のオペランドが long long 型に変換されました。

コンパイラーは両方のオペランドを unsigned long long int または unsigned long long 型に変換します。この新しい動作は、GCC コンパイラーの動作と整合します。

詳しくは、「*XL C/C++ ランゲージ・リファレンス*」の 『整数および浮動小数点数のプロモーション』を参照してください。

アーキテクチャーおよびプロセッサー・サポート

-qarch および **-qtune** コンパイラー・オプションは、コンパイラーにより生成されるコードを制御します。これらのコンパイラー・オプションは、命令、スケジューリング、およびその他の最適化を調整して、指定されたターゲット・プロセッサーまたはプロセッサーの範囲に最適なパフォーマンスが得られるようにします。

-qtune の新規デフォルト設定

新規デフォルト **-qtune** の設定は以下のとおりです。

- -qtune=balanced

-qtune=balanced サブオプションは今回のリリースからのもので、特定の **-qarch** 設定が指定された場合のデフォルトの **-qtune** 設定になります。**-qtune=balanced** を使用すると、POWER6 などの新しいプロセッサー・アーキテクチャーの範囲全体で、生成されたコードを調整してパフォーマンスを最適にするようにコンパイラーに指示します。

POWER6 プロセッサーの新規サポート

XL C/C++ バージョン 9.0 は **-qarch** および **-qtune** サブオプションのリストを拡張して、新規に使用可能な POWER6 プロセッサーをサポートします。

以下の **-qarch** および **-qtune** オプションが現在使用可能です。

- -qarch=pwr6
- -qarch=pwr6e
- -qtune=pwr6

パフォーマンスおよび最適化

パフォーマンス・チューニングおよびプログラム最適化を支援する多くの機能拡張が行われました。

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、単なる簡単な概要です。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 10. 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
-qalias= global|noglobal	これらの新規の **-qalias** サブオプションによって、リンク時の最適化中に、コンパイル単位全体で言語固有の別名割り当て規則のアプリケーションを使用可能または使用不可にすることができます。
-qalias= restrict|norestrict	これらの新規の **-qalias** サブオプションは、制限修飾ポインターの最適化を有効または無効にします。**-qalias=restrict** の指定によって、通常は制限修飾ポインターを使用するコードのパフォーマンスが向上します。**-qalias=norestrict** を使用すると、V9.0 より前のバージョンのコンパイラーでコンパイルされたコードとの互換性を保存することができます。
-qnofdpr|-qfdpr	**-qfdpr** オプションを指定すると、作成されたオブジェクト・ファイルに最適化情報を保管するように、コンパイラーに指示されます。この情報は、Feedback Directed Program Restructuring (FDPR) パフォーマンス・チューニング・ユーティリティーで使用されます。

表 10. 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 (続き)

オプション/ディレクティブ	説明
-qfloat= fenv|nofenv	これらの新規の **-qfloat** サブオプションは、浮動小数点の状況および制御レジスターの明示的な読み取りまたは書き込みのように、コードが浮動小数点ハードウェア環境で依存関係を持つかどうかをコンパイラーに通知します。 **-qfloat=nofenv** の指定は、ハードウェア環境での依存関係がないことを指示し、コンパイラーは積極的な最適化を実行することができます。
-qfloat= gcclongdouble|nogcclongdouble	これらの新しい **-qfloat** サブオプションは、 **-qldbl128** オプションが有効な場合にのみ有効です。このオプションでは、128 ビット long double 演算には GCC 提供または IBM 提供のいずれのライブラリー関数を使用するかが、コンパイラーに指示されます。
-qfloat= hscmplx|nohscmplx	**-qfloat=hscmplx** の指定によって、複素数の除法および複素数の絶対値に関連する演算の最適化が向上します。
-qfloat= rngchk|norngchk	**-qfloat=rngchk** の指定によって、ソフトウェア除算およびインラインの根号演算の入力引数の範囲検査が可能になります。 **-qfloat=norngchk** の指定は、コンパイラーに範囲検査をスキップするように指示し、特定の状況でのパフォーマンスを向上させることができます。**-qnostrict** コンパイラー・オプションの指定によって、 **-qfloat=norngchk** が設定されます。
-qipa=threads= [auto|noauto|number]	この新規の **-qipa** サブオプションを使用すると、コンパイラーが第 2 IPA パス時にコード生成に割り当てるスレッド数を指定できます。
-qnoldbl128|-qldbl128	**-qldbl128** を指定すると、long double 型のサイズが 64 ビットから 128 ビットに増大します。
-qpdf	**-qpdf** オプションを使用すると、特定のオブジェクトについての Profile-Directed Feedback を提供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 『オブジェクト・レベル Profile-Directed Feedback』 を参照してください。
-qsmp= threshold=n	**-qsmp=auto** が有効な場合、この新規サブオプションによって、コンパイラーが自動的な並列化を検討する前にループに必要な作業量を指定することができます。
#pragma expected_value(param, value)	**#pragma expected_value** ディレクティブを使用して、関数呼び出しで渡されるパラメーターが実行時に利用するのに最も適した値を指定します。コンパイラーはこの情報を使用して、関数のクローン作成やインライン化などの特定の最適化を実行することができます。

バージョン 9.0 の組み込み関数

バージョン 9.0にいくつかの組み込み関数が追加されました。

XL C/C++ によって提供される組み込み関数の詳細については、 「*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。

変換関数

次の新規関数によって、long double データ型が IBM スタイルから GCC スタイルに変換されます。

- `long double __ibm2gccldbl (long double);`
- `_Complex long double __ibm2gccldbl_cmplx (_Complex long double);`

PowerPC キャッシュ制御

PowerPC アーキテクチャーは、**dcbst** および **dcbf** キャッシュ・コピー命令を指定します。以下の新規組み込み関数によって、これらの命令に直接プログラマーがアクセスできます。

- `void __dcbst(const void* addr);  /* データ・キャッシュ・ブロック・ストア */`
- `void __dcbf(const void* addr);  /* データ・キャッシュ・ブロック・フラッシュ */`

POWER6 プリフェッチ拡張機能およびキャッシュ制御

POWER6 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のための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。

- `void __dcbfl(const void* addr);  /* pwr6 - L1 データ・キャッシュのみからのデータ・キャッシュ・ブロック・フラッシュ */`
- `void __protected_unlimited_stream_set(unsigned int direction, const void* addr, unsigned int ID);  /* pwr5 および pwr6 によりサポートされます */`
- `void __protected_unlimited_store_stream_set(unsigned int direction, const void* addr, unsigned int ID);  /* pwr6 によりサポートされます */`
- `void __protected_store_stream_set(unsigned int direction, const void* addr, unsigned int ID);  /* pwr6 によりサポートされます */`
- `void __protected_stream_count_depth(unsigned int unit_cnt, unsigned int prefetch_depth, unsigned int ID);  /* pwr6 によりサポートされます */`

その他の新規または変更されたコンパイラー・オプション

コンパイラー・オプションは、コマンド行上で、あるいはアプリケーションのソース・ファイルに組み込まれているディレクティブを通じて、指定することができます。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 11. その他の新規または変更されたコンパイラー・オプション

オプション/ディレクティブ	説明
-C!	**-C!** コンパイラー・オプションの指定によって、プリプロセスされた出力からコメントが除去されます。

表 11. その他の新規または変更されたコンパイラー・オプション (続き)

オプション/ディレクティブ	説明
-qcommon | -qnocommon	**-qcommon** が有効な場合、初期化されていないグローバル変数は、オブジェクト・ファイルの共通セクションに割り振られます。 **-qnocommon** が有効な場合、未初期化のグローバル変数は 0 に初期設定され、オブジェクト・ファイルのデータ・セクションに割り振られます。
-qoptdebug|-qnooptdebug	**-O3** 以上の最適化レベルで使用されると、新規の **-qoptdebug** オプションは、シンボリック・デバッガーが読み取ることができる最適化された疑似コードを作成するようにコンパイラーに指示します。
-qpack_semantic= ibm|gnu	**-qpack_semantic** オプションはポータビリティー・オプションで、 **#pragma pack** ディレクティブに IBM または GCC のいずれの構文およびセマンティクスを使用するかがコンパイラーに指示されます。
-qreport	**-qreport** オプションは、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、プログラム・ループがどのように並列化され、ベクトル化されるかを示す疑似コード・リストを作成します。 このレポートも、コンパイラーが指定されたループを並列化またはベクトル化することができない場合は、診断情報を提供します。
-qsaveopt|-qnosaveopt	以前のリリースでは、 **-qsaveopt** オプションは、ファイルをコンパイルするのに使用したコマンド行オプションを結果のオブジェクト・ファイルに保管していました。 バージョン 9.0 では、オブジェクト・ファイルに保管される情報が、コンパイル時に呼び出された各コンパイラー・コンポーネントのバージョンおよびレベル情報も含むように拡張されました。
-qsmp=stackcheck	この新規の **-qsmp** サブオプションは、実行時にスレーブ・スレッドによってスタック・オーバーフローの有無を確認し、残りのスタック・サイズが XLSMPOPTS 環境変数の **stackcheck** オプションに指定されたバイト数に満たない場合は警告を出すようにコンパイラーに指示します。
-qtemplatedepth=number	**-qtemplatedepth** は、コンパイラーが処理する、再帰的にインスタント化されるテンプレートの特殊化の最大数を指定します。
-qversion=verbose	**-qversion** オプションは、新規の **verbose** サブオプションを追加しています。 **-qversion=verbose** の指定は、コンパイル時に呼び出されたコンパイラー・コンポーネントのバージョンおよびレベル情報を表示するようにコンパイラーに指示します。

第 4 章 XL C/C++ のセットアップとカスタマイズ

XL C/C++ に関する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『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

コンパイラーの設定およびオプションのカスタマイズは、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することによって行うことができます。

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とともに、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して解決することができます。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 (デフォルトではコンパイルと結合された単一ステップ)、 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。

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。
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。リンク時の最適化については、一部の活動がコンパイル中に複数回実行されます。各々のプログラムが実行されるときに、結果が、順序に並んでいるものの中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++ が認識する接尾部のリストについては、 44 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。

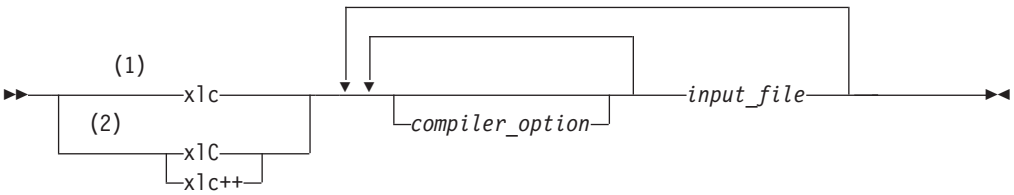
XL C/C++ によるコンパイル

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。

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



注:

1 C ソース・コードをコンパイルするための基本呼び出し

2 C++ ソース・コードをコンパイルするための基本呼び出し

大部分のアプリケーションでは、**xlc**、**xlc++**、またはスレッド・セーフな対応物を指定してコンパイルする必要があります。ユーザーは **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。

追加の呼び出しコマンドは、特殊なコンパイル要件を満たすために使用でき、主として、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・サポートを提供します。 GNU コンパイル環境からXL C/C++ に移行する開発者を支援するための特殊な呼び出しを含む、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++ では、マルチプロセッサー環境で使用するための並列化アプリケーションをコンパイルする際に使用できるスレッド・セーフのコンパイラー呼び出しコマンドを使用できます。

これらの呼び出しは、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。以下は、汎用 XL C/C++ スレッド・セーフ・コンパイラー呼び出しです。

- 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・オプションは以下のようにユーザーが指定できます。

- 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
- または、これらの手法を任意に組み合わせたものを使用して

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、上記の優先順位に従わないものもあります。

例えば、**-I** コンパイラー・オプションは特別なケースです。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、vac.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。これらのオプションは、優先的ではなく、累積的です。

コンパイラー・オプションとその使用法に関する詳細については、「*XL C/C++ コンパイラー・リファレンス*」を参照してください。

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。

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・オプションおよび指定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・オプション・リファレンス』 を参照してください。

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++ コマンドを含む、GNU C/C++ コンパイラーから XL C/C++ への移行に役立つさまざまな機能が含まれています。

gxlc および gxlc++ ユーティリティーはそれぞれ GNU C または C++ コンパイラー・オプションを受け入れて、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++** 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++ の機能を十分に活用するためには、XL C/C++ 呼び出しコマンドおよびその関連オプションを使用する必要があります。

gxlc および gxlc++ のアクションは、構成ファイル `gxlc.cfg` 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++ オプションの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*」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・タイプに関する詳細については、 「*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 12. 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 13. 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、以下のコマンドを実行すると、

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

`file1.C` および `file3.C` がコンパイルされてオブジェクト・ファイル `file1.o` および `file3.o` が作成され、次にすべてのオブジェクト・ファイル (`file2.o` も含まれる) がリンカーに処理依頼されて 1 つの実行可能モジュールが作成されます。

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
  # 1 つのオブジェクト・ファイル (file1.o) を作成する
xlc++ -c file2.C file3.C
  # または複数のオブジェクト・ファイル (file1.o、file3.o) を作成する
xlc++ file1.o file2.o file3.o
  # オブジェクト・ファイルをデフォルト・ライブラリーとリンクする
```

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きに、共用ライブラリーはデフォルトで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ると、使用するディスク・スペースと仮想メモリーが少なくて済みます。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用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実行する場合があります。また、それらを使用すれば、再リンクしないで共用ライブラリー内のルーチンをアップグレードすることができます。

このリンク形式はデフォルトなので、これを作動させ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・ルーチンへの呼び出しを多数行ったり、多数の小さなルーチンを呼び出す場合、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実行する場合があります。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オペレーティング・システムのあるレベルでコンパイルした後、そのオペレーティング・システムの別のレベルで実行すると、それらは機能しない場合があります。

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。

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。

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。

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。 他の環境変数は、実際のランタイムの動作を制御しませんが、アプリケーションの実行の仕方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作成し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したい場合には、そのシステムにランタイム環境をインストールする必要があります。

最新の XL C/C++ Runtime Environment PTF イメージを、ライセンス交付および使用に関する情報と一緒に、次のサイトにある XL C/C++ Support ページから取得することができます。

www.ibm.com/software/awdtools/xlcpp/support

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** または **-qlinedebug** コンパイラー・オプションを指定することにより、XL C/C++ コンパイラーに、コンパイルされた出力にデバッグ情報を組み込むように指示できます。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・オプション』を参照してください。

次に、AIX XCOFF 実行可能形式をサポートする **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て、検査できます。

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。

どのレベルの XL C/C++ がインストールされているかの判別

ソフトウェア・サポートに連絡して支援を得るには、特定のマシンにインストールした XL C/C++ のレベルを確認する必要があります。

ご使用のシステムにインストールされたコンパイラーのバージョンおよびリリース・レベルを表示するために、**-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、バージョン情報の詳細を表示するには、コマンド行に以下を入力します。

```
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。

〒242-8502
神奈川県大和市下鶴間1623番14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。

商標およびサービス・マーク

IBM、IBM ロゴ、および ibm.com は、International Business Machines Corporation の米国およびその他の国における商標です。この資料が公開された時点で、米国において IBM が所有する登録商標または商標には、初出時に商標記号 (® または ™) を付けて示しています。このような商標は、その他の国においても、登録商標または商標である可能性があります。現時点での IBM の商標については、http://www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript 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。

Microsoft、Windows は、Microsoft Corporation の米国およびその他の国における商標です。

Cell Broadband Engine, Cell/B.E は、米国およびその他の国における Sony Computer Entertainment, Inc. の商標であり、同社の許諾を受けて使用しています。

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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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

V

X

［特殊文字］

プログラム番号: 5724-X14

Printed in Japan

GI88-4231-00

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