IBM XL C/C++ Advanced Edition for Linux, V9.0

XL C/C++ はじめに

GC88-4645-00
(英文原典：GC23-5891-00)

IBM XL C/C++ Advanced Edition for Linux, V9.0

XL C/C++ はじめに

GC88-4645-00
(英文原典：GC23-5891-00)

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、 25 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++ Advanced Edition for Linux, V9.0 (プログラム番号 5724-S73) および新しい版で特に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。必ず、製品のレベルに合った正しい版を使用してください。

IBM 発行のマニュアルに関する情報のページ

http://www.ibm.com/jp/manuals/

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

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： GC23–5891–00
IBM XL C/C++ Advanced Edition for Linux, V9.0
Getting Started with XL C/C++

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2007.6

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について

本書には、 IBM® XL C/C++ Advanced Edition for Linux® V9.0 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記載されています。

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++ コンパイラー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。

本書の使用法

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、『本書で使用する規則』 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++ Installation Guide*」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++ Compiler Reference*」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 C または C++ プログラミング言語の IBM インプリメンテーションの詳細については、「*XL C/C++ Language Reference*」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++ Programming Guide*」を参照してください。

本書で使用する規則

活字規則

下記の表では本書で使用する活字規則を説明します。

表 1. 活字規則

字体	指示	例
太字	小文字のコマンド、実行可能プログラム名、コンパイラー・オプションおよびディレクティブ。	**-O3** を指定すると、コンパイラーは **-qhot=level=0** を前提とします。 **-O3** のすべての HOT 最適化を抑制するには、**-qnohot** を指定する必要があります。
イタリック	実際の名前や値がユーザーによって供給される、パラメーターまたは変数を識別します。イタリック体は新規用語を紹介するためにも使用されます。	要求を超えるサイズ を戻す場合は、*size* パラメーターを更新する必要があります。
モノスペース	プログラミング・キーワードとライブラリー関数、コンパイラー組み込み機能、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。	`switch` 文の 1 つか 2 つのケースが一般的に他のケースより頻繁に実行される場合は、`switch` 文の前に別々に処理して、これらのケースを抜き出します。

アイコン

本書内に記載されるすべての機能は、C および C++ 言語に適用されます。ある機能が 1 つの言語に専用の場合、または機能性が言語によって異なる場合は、以下のアイコンが使用されます。

テキストは C 言語のみによってサポートされるフィーチャーを説明します。または C 言語のみに特定な動作を説明します。

テキストは C++ 言語のみによってサポートされるフィーチャーを説明します。または C++ 言語のみに特定な動作を説明します。

構文図

本書では、XL C/C++ の構文を構文図で示しています。このセクションでは、構文図の見方と使い方について説明します。

- 構文図は、左から右、上から下に、線のパスに従って読んでください。

  ▸▸── 記号は、コマンド、ディレクティブ、または文の開始を示します。

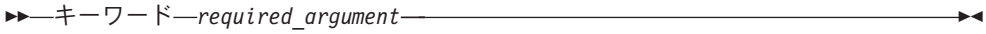
  ──▸ 記号は、コマンド、ディレクティブ、または文構文が次の行に続いていることを示します。

  ▸── 記号は、コマンド、ディレクティブ、または文構文が前のライン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、または文の終了を示します。

  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図である断片は、|── 記号で開始して、──| 記号で終了します。

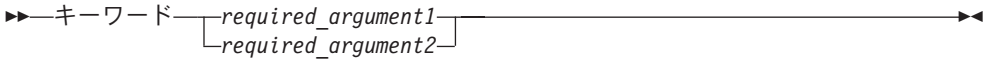
- 必須項目は水平線 (メインパス) 上に示されています。



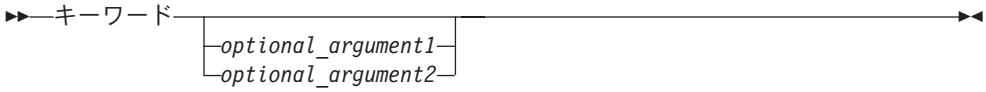
- オプション項目は、メインパスの下側に表示されます。

- 2 項目以上から選択できる場合は、縦に積み重ねて表示されます。

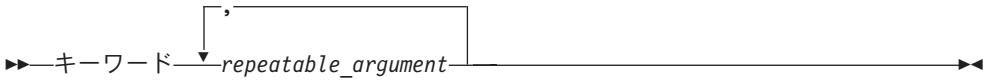
  1 項目の選択が必要 な場合は、スタックの 1 項目がメインパスに表示されます。



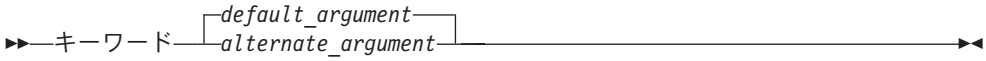
  項目の 1 つの選択がオプショナルの場合は、全体のスタックがメインパスの下側に表示されます。



- メインラインの左上に戻る矢印 (繰り返し矢印) は、スタックされた項目から複数を選択できるか、同じ項目を繰り返し選択できることを示します。区切り文字も、ブランク以外であれば示されています。



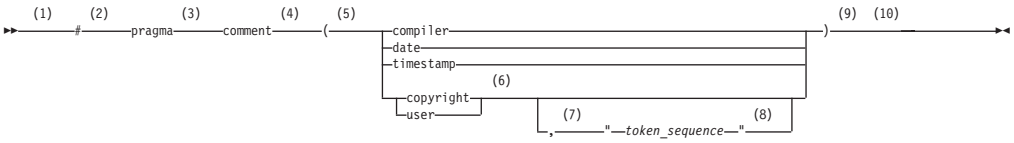
- デフォルトである項目はメインパスの上側に表示されます。



- キーワードは非イタリック体で表示され、表示されたとおり正確に入力されなければなりません。
- 変数はイタリック体を使用した小文字で表示されます。変数はユーザーが指定する名前または値を表します。
- 句読記号、括弧、算術演算子、または他の示されている記号は、構文の一部としてユーザーが入力する必要があります。

構文図の例

下記の構文図の例は、**#pragma comment** ディレクティブの構文を示します。



注:

1 これは構文図の開始です。

2 記号 # は先頭になければなりません。

3 キーワード pragma は # 記号の後になければなりません。

4 キーワード pragma の後にはプラグマの名前 comment が必要です。

5 左括弧が必要です。

6 コメント・タイプは、示されたタイプの中の 1 つ、つまり compiler、date、timestamp、copyright、または user としてのみ入力できます。

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 コンマの後には文字ストリングが必要です。文字ストリングは二重引用符で囲まれなければなりません。

9 右括弧が必要です。

10 これは構文図の終了です。

下記の **#pragma comment** ディレクティブの例は、上記の図に従えば構文的には正しい表示です。

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

例

この文書の例は、特に注記がない限り単純なスタイルでコード化されていて、ストレージを節約し、エラーのチェック、高速なパフォーマンス、または特定の結果を達成するためにすべての可能性のあるメソッドを実証するようなことは行いません。

関連情報

以下のセクションでは、XL C/C++ 関連の資料に関する情報を提供します。

- 『IBM XL C/C++ の資料』
- x ページの『その他の IBM 資料』
- x ページの『その他の資料』

IBM XL C/C++ の資料

XL C/C++ には、下記の形式の製品資料があります。

- README ファイル

README ファイルには、製品資料についての変更や訂正を含む最新情報が入っています。 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・ディレクトリーに入っています。

- インストール可能マニュアル・ページ

  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については、マニュアル・ページが製品に付属して提供されています。マニュアル・ページのインストールおよびアクセスの説明については、「*XL C/C++ Installation Guide*」を参照してください。

- インフォメーション・センター

  検索可能な HTML ファイルのあるインフォメーション・センターをネットワークに起動して、リモートまたはローカル側からアクセスできます。オンライン・インフォメーション・センターのインストールおよびアクセスの説明については、「*XL C/C++ Installation Guide*」を参照してください。インフォメーション・センターは、次の Web サイトからも表示できます。http://publib.boulder.ibm.com/infocenter/lnxphelp/v9v111/index.jsp

- PDF 文書

  PDF 文書は、デフォルトでは /opt/ibmcmp/vacpp/9.0/doc/*LANG*/pdf/ ディレクトリーに入っています。ここで、*LANG* は en_US、zh_CN、 または ja_JP です。PDF ファイルは Web からも入手できます (http://www.ibm.com/software/awdtools/xlcpp/library)。

  XL C/C++ 製品マニュアルは、すべて下記のファイルに収められています。

表 2. 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++ Advanced Edition for Linux, V9.0 インストール・ガイド, GC88-4647-00	install.pdf	XL C/C++ のインストールおよび基本的なコンパイルとプログラム実行に必要な環境の構成に関する情報が含まれています。
IBM XL C/C++ Advanced Edition for Linux, V9.0 はじめに, GC88-4645-00	getstart.pdf	XL C/C++ 製品の入門が用意されていて、お客様の環境の設定と構成に関する情報、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のための情報など含まれています。
IBM XL C/C++ Advanced Edition for Linux, V9.0 Compiler Reference, SC23-5889-00	compiler.pdf	各種のコンパイラー・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 (並列処理用のものも含む) に関する情報が含まれています。
IBM XL C/C++ Advanced Edition for Linux, V9.0 Language Reference, SC23-5892-00	langref.pdf	IBM がサポートする C および C++ プログラミング言語 (所有権のない標準に対する移植性と適合性のための言語拡張を含む) に関する情報が含まれています。

表 2. XL C/C++ PDF ファイル (続き)

文書タイトル	PDF ファイル名	説明
IBM XL C/C++ Advanced Edition for Linux, V9.0 プログラミング・ガイド, SC88-4646-00	proguide.pdf	アプリケーションの移植、Fortran コードを使った言語間呼び出し、ライブラリー開発、アプリケーションの最適化と並列化、および XL C/C++ ハイパフォーマンス・ライブラリーといった、高度なプログラミング・トピックに関する情報が含まれています。

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、(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。

Redbook、ホワイト・ペーパー、チュートリアルその他の文献を含む XL C/C++ 関連の豊富な資料を下記の Web サイトから入手できます。

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

その他の IBM 資料

- *ESSL for Linux on POWER V4.2 Guide and Reference*, SA22-7904、http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp で入手可能です。

その他の資料

- *Using the GNU Compiler Collection*、http://gcc.gnu.org/onlinedocs で入手可能です。

テクニカル・サポート

追加のテクニカル・サポートは、XL C/C++ サポート・ページ (http://www.ibm.com/software/awdtools/xlcpp/support) から入手できます。このページでは、さまざまなテクニカル・サポート FAQ その他のサポート文書に対する検索機能を備えたポータルを提供します。

もし、見つからない場合は、compinfo@ca.ibm.com 宛に電子メールをご送付ください。

XL C/C++ に関する最新の情報については、製品情報サイト (http://www.ibm.com/software/awdtools/xlcpp)をご覧ください。

第 1 章 XL C/C++ の紹介

IBM XL C/C++ Advanced Edition for Linux, V9.0 は高機能で高性能なコンパイラーであり、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

XL C/C++ は、XL C および 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。

IBM XL C/C++ Advanced Edition for Linux, V9.0 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。

これらのコンパイラーは、AIX®、 i5/OS®、選択された Linux ディストリビューション、 z/OS®、および z/VM® オペレーティング・システムなどの、多様なプラットフォームおよびプログラミング言語でのコンパイラーの関数および最適化テクノロジーを共有する共通のコード・ベースから導き出されています。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

XL C/C++ の V9.0 は、複数の Linux ディストリビューション。 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++ Installation Guide*」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 POWER3™、POWER4™、 POWER5™、 POWER5+™、 POWER6™、PowerPC®、および PowerPC 970 システム上で実行することができます。

サポートされているさまざまなハードウェア構成を最大限に活用するために、コンパイラーには、特にコンパイルされ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性能チューニング・アプリケーションの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

XL C/C++ は、ユーザーの独自の固有のコンパイル要件に合わせてユーザーがコンパイラーを調整できるようにする豊富な機能を提供しています。

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++ Compiler Reference*」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拡張機能を選択し、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++ Compiler Reference*」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。

詳細については、 15 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。

言語標準への準拠

コンパイラーは、 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機能もサポートします。

- 並列化されたプログラミングをサポートする OpenMP V2.5 拡張機能
- ベクトル・プログラミングをサポートする言語拡張機能
- GNU C および C++ 言語拡張機能のサブセット

GNU との互換性

XL C/C++ は、**gcc**および **g++** 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。詳細については、「*XL C/C++ Compiler Reference*」の中の『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、コンパイラーにユーザーのアプリケーションをコンパイルするように指示するコンパイラー呼び出しコマンドを提供することによって、ユーザーの既存の ソース・コードに対する投資を保護するのに役立ちます。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・プログラムのソース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合致しない場合、警告、エラー、および重大エラー・メッセージを出します。

詳細については、「*XL C/C++ Compiler Reference*」の『**-qlanglvl**』を参照してください。

ライブラリー

XL C/C++ は、以下のライブラリーとともに出荷されます。

- SMP Runtime Library は、自動化され、かつ明示的な並列処理をサポートします。
- 32 ビット・モードおよび 64 ビット・モード用の、調整された数学組み込み関数の Mathematical Acceleration Subsystem (MASS) ライブラリー。
- 調整された代数関数の Basic Linear Algebra Subprograms (BLAS) ライブラリー。
- C++ 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。

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。 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。

詳細については、「*XL C/C++ Programming Guide*」の中の『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出荷されます。 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++ Programming Guide*」の中の『Basic Linear Algebra Subprograms の使用』を参照してください。

ツールおよびユーティリティー

new_install

IBM XL C/C++ Advanced Edition for Linux, V9.0 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、ご自身で可能な追加の構成ファイルの作成を行い、次に、コンパイラー・オプション・デフォルト設定値のご自身のカスタム・セットを入れるように変更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能力を提供します。 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 コマンドの現行の動作は、**resetpdf** コマンドと同じであり、これは他プラットフォーム上の以前のリリースとの互換性を備えておくために保持されています。

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のカウントを表示します。

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++ Advanced Edition for Linux, V9.0 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++ Advanced Edition for Linux, V9.0 への変換を容易にすることにあります。

プログラムの最適化

XL C/C++ は、ユーザーのプログラムの最適化を制御するのに役立つことのでき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、以下に挙げることを行えます。

- 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。 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、以下に挙げることを行えます。

- クリティカルな操作に対して実行する命令の数を減らす。
- 生成されたオブジェクト・コードを再編成して、PowerPC アーキテクチャーの使用を最適化する。
- メモリー・サブシステムの使用を向上させる。
- 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を活用する。

コンパイラーは、洗練されたプログラムの分析および変換が可能であるため、開発時には比較的少ない労力しかけずに、パフォーマンスを大幅に向上させることができます。さらに、XL C/C++ は OpenMP などのプログラミング・モデルを使用可能にし、それにより、ユーザーはハイパフォーマンスの並列コードを作成できるようになります。

詳細については、以下を参照してください。

- 「*XL C/C++ Programming Guide*」の『アプリケーションの最適化』
- 「*XL C/C++ Compiler Reference*」の『最適化およびチューニング・オプション』
- 「*XL C/C++ Compiler Reference*」の中の『コンパイラー組み込み機能』

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、主として **-q64** 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。

詳細については、「*XL C/C++ Programming Guide*」の『32 ビットおよび 64 ビットのモードの使用』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)

詳細については、「*XL C/C++ Programming Guide*」の中の『プログラムの並列化』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 SMP 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++ は、OpenMP API バージョン 2.5 仕様をサポートします。

詳細については、以下を参照してください。

- 「*XL C/C++ Programming Guide*」の中の『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には、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提供されていることがあります。

リストされる情報には、組み込むことも、あるいは省略することもできるオプションの部分があります。適用可能なコンパイラー・オプションおよびそのリスト作成について詳しくは、「*XL C/C++ Compiler Reference*」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。

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。その情報は、**gdb** または 他の任意のシンボリック・デバッガーを使用して調べることができ、ユーザーのプログラムのデバッグに役立ちます。

第 2 章 IBM XL C/C++ Advanced Edition for Linux, V9.0 の新機能

このセクションでは、IBM XL C/C++ Advanced Edition for Linux, V9.0 の新機能および機能拡張について説明します。

C/C++ 言語関連の更新

このリリースでは、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追加されました。

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現在、**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** 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することなく、 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 __STD_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

本リリースの XL C/C++ V9.0 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 型に変換されます。

本リリース以降、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++ Language Reference*」の『整数および浮動小数点の上位変換』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++ 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

-qipa コンパイラー・オプションも、新規のアーキテクチャー・クローン作成サブオプションを追加して、POWER6 プロセッサーでのプロシージャー間分析 (IPA) の最適化をサポートします。

- -qipa=clonearch=pwr6

パフォーマンスおよび最適化

新機能と機能拡張の多くは、パフォーマンスおよび最適化チューニングのカテゴリーに分類されます。

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++ Compiler Reference*」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 3. 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。
-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	この新規 **-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=clonearch=pwr6	**-qipa=clonearch** コンパイラー・オプションには現在、新規の **pwr6** サブオプションが組み込まれ、 POWER6 プロセッサーにおけるプロシージャー間分析 (IPA) の最適化をサポートします。
-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。

表 3. 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
-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。詳細については、「*XL C/C++ Programming Guide*」の『オブジェクト・レベルのプロファイル指示フィードバック』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。

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の詳細については、「*XL C/C++ Compiler Reference*」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++ Compiler Reference*」を参照してください。

表 4. 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。
-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** が有効の場合、初期化されていないグローバル変数は、ゼロに初期化されて、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・リストを作成するようになりました。 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。 今回のリリースでは、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。

表 4. 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
-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 **stackckeck** 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 3 章 XL C/C++ のセットアップとカスタマイズ

前提条件およびインストールに関する完全な情報については、「*XL C/C++ Installation Guide*」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

デフォルトのコンパイラー構成ファイルは、XL C/C++ コンパイラーのインストール中に作成されます。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も再適用する必要があります。

これを避けるために、独自のカスタム・コンパイラー構成ファイルを作成することができます。コンパイラーは現在、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したコンパイラー設定を認識して解決する能力を持っています。

カスタム構成ファイルを使用するようコンパイラーに指定した場合、コンパイラーは、デフォルトのシステム構成ファイル内の設定を確認する前に、カスタム構成ファイル内の設定を調べて処理します。後に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でも、カスタム構成ファイル内の設定には影響を与えません。

詳しくは、「*XL C/C++ Compiler Reference*」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。

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
```

第 4 章 XL C/C++ によるアプリケーションの開発

基本的な アプリケーション開発は、編集、コンパイル、リンク (デフォルトでコンパイルと組み合わせた単一ステップ)、および実行の反復されるサイクルで構成されています。

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。 詳細については、「*XL C/C++ Installation Guide*」を参照してください。
2. プログラムの作成について学習するには、「*XL C/C++ Language Reference*」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. プリプロセス後の **.s** ファイルおよび **.S** ファイルのプログラム・アセンブリー
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。

ソース・ファイルの編集

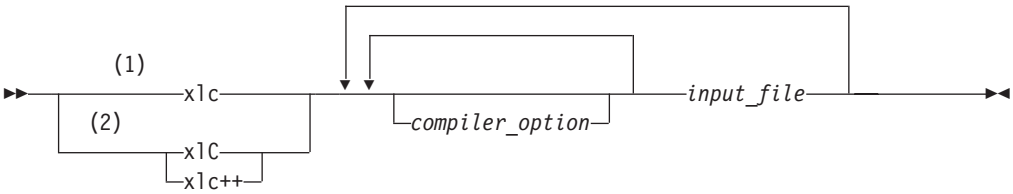
ソース・プログラムを作成するには、ご使用のシステムで使用可能な任意のテキスト・エディターを使用することができます。 ソース・プログラムは、認識されたファイル名接尾部を使用して保管する必要があります。 XL C/C++ が認識する接尾部のリストについては、 20 ページの『XL C/C++ 入力および出力ファイル』を参照してください。

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++ Language Reference*」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

コンパイラーの呼び出し

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

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。

新しい C または C++ アプリケーションの作業では、 **xlC**、 **xlc++**、 またはスレッド・セーフな対応のものを指定してコンパイルする必要があります。 **xlC** と **xlc++** は共に、 C または C++ プログラム・ソースをコンパイルしますが、**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と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援助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++ Compiler Reference*」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++ は、マルチプロセッサー環境で使用される並列化アプリケーションをコンパイルするの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提供します。 これらの呼び出しは、コンパイル済みの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以外は、対応する基本コンパイラー呼び出しに似ています。

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に、追加スレッド・セーフ呼び出しを提供します。 詳細については、「*XL C/C++ Compiler Reference*」の中の『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 つと一緒に **-qsmp** オプションのみを指定すればいいということです。**-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。

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++ Compiler Reference*」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・オプションおよびその指定方法についての詳細は、「*XL C/C++ Compiler Reference*」の『コンパイラー・オプションの参照』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、 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。 それらの機能として、gxlc および gxlc++ ユーティリティーがあります。

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++** 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・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関する情報については、「*XL C/C++ Compiler Reference*」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

以下にリストされているファイル・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は、「*XL C/C++ Compiler Reference*」の『入力ファイルのタイプ』 および『出力ファイルのタイプ』を参照してください。

表 5. 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

表 5. 入力ファイル・タイプ (続き)

ファイル名拡張子	説明
.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 6. 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
```

オブジェクト・ファイル `file1.o` および `file3.o` がコンパイルされ、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
```

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、XL C/C++ ランタイム・ライブラリーがないシステムに移動してその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。

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++ Compiler Reference*」の『リンク』
- 「*XL C/C++ Programming Guide*」の『ライブラリーの構成』

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・オプションを指定すれば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 (例えば **test** または **cp** といった) システムまたはシェルのコマンドと同じ

名前を付けることは回避する必要があります。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、プログラム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。

ランタイム・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ランタイム・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++ Installation Guide*」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。

XL C/C++ ランタイムもまた、アプリケーションの実行時に、特に入出力関連のサポートされていない特定の操作に対してメッセージを出します。

リスト作成、診断、アプリケーションの問題の解決に役立つ関連コンパイラー・オプションについて詳しくは、「*XL C/C++ Compiler Reference*」の中にある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

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は、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 XL C/C++ コンパイラーに指示することになります。

そして、**<gdb** などのシンボリック・デバッガーを使用して、コンパイル済みのアプリケーションの動作をステップスルーおよび検査できます。

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。 デバッグに関する詳細については、「*XL C/C++ Programming Guide*」の 『アプリケーションの最適化』を参照してください。

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。

〒106-8711
東京都港区六本木 3-2-12
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

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

資料に記載されている会社名、製品名、またはサービス名は、IBM Corporation または各社の商標である可能性があります。IBM Corporation の商標については、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

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

Intel は、Intel Corporation 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。

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。

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。

業界標準

以下の標準がサポートされています。

- C 言語は、Information Systems-Programming Language C の国際標準 (ISO/IEC 9899-1990) に整合したものです。
- C 言語は、Information Systems-Programming Language C の国際標準 (ISO/IEC 9899-1999 (E)) にも整合したものです。
- C++ 言語は、Information Systems-Programming Language C++ の国際標準 (ISO/IEC 14882:1998) に整合したものです。
- C++ 言語は、Information Systems-Programming Language C++ の国際標準 (ISO/IEC 14882:2003 (E)) にも整合したものです。
- C および C++ 言語は、 OpenMP C および C++ Application Programming Interface Version 2.5 に整合したものです。

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]

G

M

O

S

V

X

［特殊文字］

プログラム番号: 5724-S73

GC88-4645-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12