IBM XL C/C++ Advanced Edition V8.0 for Linux

XL C/C++ 入門

SD88-6730-00
(英文原典：SC09-8015-00)

IBM XL C/C++ Advanced Edition V8.0 for Linux

XL C/C++ 入門

SD88-6730-00
(英文原典：SC09-8015-00)

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、 39 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 V8.0 for Linux™ (プログラム番号 5724-M16) および、新版で特に断りがない限り後続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。製品のレベルに合った正しい版をご使用になるようお確かめください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(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： SC09–8015–00
IBM XL C/C++ Advanced Edition V8.0 for Linux
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 2005.9

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 v
本書の対象読者 v
本書の使用法 v
本書の構成 v
本書で使用される規則および用語 vi
書体の規則 vi
構文図の読み方 vi
例 viii
関連情報 viii
IBM XL C/C++ の資料 viii
追加資料 x
技術サポート x

第 1 章 XL C/C++ の機能の概要 1
他の XL コンパイラーとの共通点 1
資料、オンライン・ヘルプ、および技術サポート . . 1
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 2
高度に構成することが可能なコンパイラー 2
言語標準への準拠 3
GNU との互換性 4
ソース・コード・マイグレーションおよび規格合致検査 4
ライブラリー 4
Mathematics Acceleration Subsystem ライブラリー 5
Basic Linear Algebra Subprogram 5
ツールおよびユーティリティー 5
プログラムの最適化 6
64 ビット・オブジェクトの機能 7
共用メモリーの並列処理 8
OpenMP ディレクティブ 8
診断リスト 8
シンボリック・デバッガー・サポート 9

第 2 章 V8.0 の新機能 11
パフォーマンスおよび最適化 11
アーキテクチャーおよびプロセッサー固有のコードのチューニング 11
高性能ライブラリー 11
他のパフォーマンス関連コンパイラー・オプションおよびディレクティブ 12
このリリースでの新規組み込み関数 13
言語の機能拡張および API に対するサポート . . . 16
C、C++、および Fortran に対する OpenMP API V2.5 サポート 16
使いやすさ 16
新規のインストールおよび構成ユーティリティー 16
IBM Tivoli License Manager のサポート 16
新規のコンパイラー・オプション 17
新規のコマンド行オプション 17
新規のプラグマ・ディレクティブ 18

第 3 章 XL C/C++ のセットアップとカスタマイズ 21
環境変数と XL C/C++ 21
コンパイラー作動環境の設定 21
デフォルトのランタイム・オプションの設定 . . 21
構成ファイルのカスタマイズ 22
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 22

第 4 章 XL C/C++ でのプログラムの編集、コンパイル、およびリンク 23
コンパイラー・フェーズ 23
C および C++ ソース・ファイルの編集 23
XL C/C++ によるコンパイル 24
並列化 XL C/C++ アプリケーションのコンパイル 25
XL C/C++ 入力ファイル 26
XL C/C++ 出力ファイル 27
コンパイラー・オプションの指定 28
XL C/C++ プログラムのリンク 29
別個のステップでのコンパイルおよびリンク . . 29
動的および静的リンク 30

第 5 章 XL C/C++ プログラムの実行 . . 31
実行の取り消し 31
ランタイム・オプションの設定 31
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 31

第 6 章 XL C/C++ コンパイラー診断エイド 33
コンパイルおよび戻りコード 33
XL C/C++ コンパイラー・リスト 34
ヘッダー・セクション 34
オプション・セクション 35
ソース・セクション 35
変換レポート・セクション 37
属性および相互参照セクション 37
オブジェクト・セクション 37
ファイル・テーブル・セクション 38
コンパイル単位エピローグ・セクション 38
コンパイル・エピローグ・セクション 38
コンパイル済みアプリケーションのデバッグ . . . 38

特記事項 39
プログラミング・インターフェース情報 40
商標 41

索引 43

本書について

「*XL C/C++ 入門*」XL C/C++ コンパイラーの全般的概要、その重要度の高い機能、およびそれらの機能がお客様のソフトウェア開発の生産性向上にどのように役立つかについて記載しています。

このリリースにアップグレードしようとしている現行の XL C/C++ ユーザーの便宜のために、「*XL C/C++ 入門*」には、V8.0 での新規機能または改良された機能の要約も記載されています。

「*XL C/C++ 入門*」は、単にコンパイラーに慣れていただくのに役立つことだけを意図しています。 XL C/C++ コンパイラーを使用する上での詳細な情報については、 viii ページの『IBM XL C/C++ の資料』に説明する XL C/C++ Advanced Edition V8.0 for Linux 資料ライブラリーにある他の資料を参照してください。

本書の対象読者

「*XL C/C++ 入門*」は、IBM® XL C/C++ Advanced Edition V8.0 for Linux を扱う仕事をするよう計画している方、 Linux® オペレーティング・システムに精通している方、および C および C++ のプログラミングを多少とも経験したことのある方を対象に書かれたものです。

本書の使用法

まだ XL C/C++ に慣れていない場合には、 1 ページの『第 1 章 XL C/C++ の機能の概要』に目を通して、XL C/C++ の主要な機能、およびご自身のアプリケーションを開発するためにこれを使用し始める方法に習熟してください。

すでに経験を積んだ XL C/C++ のユーザーであり、XL C/C++ の最新リリースにアップグレードする場合には、直接 11 ページの『第 2 章 V8.0 の新機能』に進み、コンパイラーに対する最新の変更と機能拡張を調べてください。

このガイドのその他の部分では、XL C/C++ での基本的なプログラム開発タスクの簡潔な概要を記載しています。

本書の構成

このガイドには、以下のトピックが含まれています。

- 1 ページの『第 1 章 XL C/C++ の機能の概要』は、XL C/C++ コンパイラーの主要な機能の概要を述べています。
- 11 ページの『第 2 章 V8.0 の新機能』は、XL C/C++ の最新バージョンによって提供されている新規および更新された機能を説明しています。
- 21 ページの『第 3 章 XL C/C++ のセットアップとカスタマイズ』は、XL C/C++ をセットアップしカスタマイズするのに関係する手順についての簡潔な概要の情報、および、より詳細な情報の所在を示す情報を提供します。

- 23 ページの『第 4 章 XL C/C++ でのプログラムの編集、コンパイル、およびリンク』は、XL C/C++ を使用してユーザーのアプリケーションを作成し、コンパイルするのに関係する基本的な手順について述べます。
- 31 ページの『第 5 章 XL C/C++ プログラムの実行』は、ランタイム・オプションの設定も含めて、コンパイルされたアプリケーションを実行する方法を説明します。
- 33 ページの『第 6 章 XL C/C++ コンパイラー診断エイド』は、XL C/C++ コンパイラー診断エイドを使用して、ユーザーのアプリケーションに関するコンパイル上の問題を識別し訂正する方法を提供します。

本書で使用される規則および用語

書体の規則

次の表は、本書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	コマンド、実行可能名、 プラグマ・ディレクティブ、およびコンパイラー・オプション。	**-qmkshrobj** コンパイラー・オプションを使用して、生成されたオブジェクト・ファイルから共用オブジェクトを作成します。
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よう確認してください。
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、ファイル名およびディレクトリー名、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。	未初期化ロック変数を用いて `omp_destroy_lock` を呼び出すと、その呼び出しの結果は未定義になります。

構文図の読み方

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。

句読記号、括弧、算術演算子、およびその他の特殊文字を、構文の一部として入力する必要があります。

- 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。

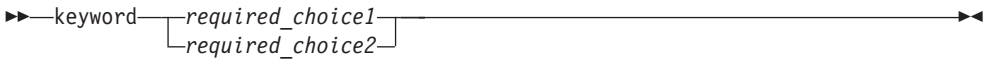
完結したコマンド、ディレクティブ、またはステートメント以外の構文単位の図は、►— で始まり、⟶► 記号で終わります。

- 必須項目は、水平線 (主経路) 上に示されます。

- オプション項目は、主経路の下に示されます。

- 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。

  複数の項目の中から 1 つを選択する必要がある 場合には、スタックの 1 つの項目が主経路上に示されます。



  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。

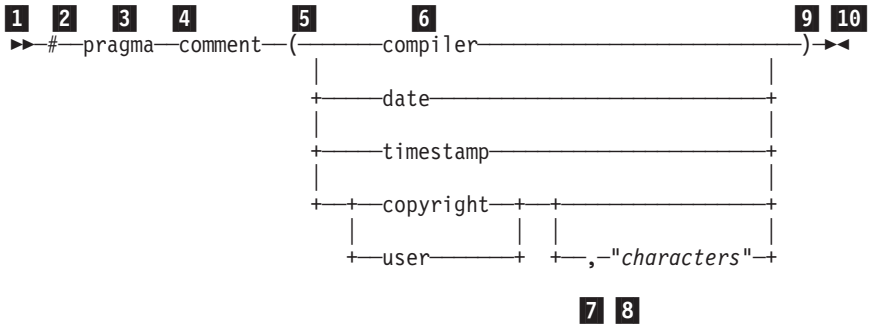
  デフォルトである項目は、主経路の上に示されます。

- 主線の上にあって左へ戻る矢印は、繰り返すことができる項目を示します。

  スタックの上の反復矢印は、スタックされた項目から複数個選択できること、あるいは単一項目の選択を繰り返すことができることを示します。
- 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 (たとえば `extern`) 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 (たとえば *identifier*) 示されます。変数は、ユーザーが提供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。

次の構文図の例は、**#pragma comment** ディレクティブの構文を示しています。



1 これが構文図の始まりです。

2 記号 # が最初に示される必要があります。

3 キーワード pragma が # 記号の次に示される必要があります。

4 プラグマの名前 comment が、キーワード pragma の次に示される必要があります。

5 左括弧がなければなりません。

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

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

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

9 右括弧が必要です。

10 これが、構文図の終わりです。

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

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

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、エラーを検査する、速いパフォーマンスを達成する、または特定の結果を得るための可能なあらゆる方法を示す、といったことを試みることは行わない、単純な形式でコーディングしてあります。

関連情報

IBM XL C/C++ の資料

XL C/C++ は、製品資料を、以下の形式で提供します。

- README ファイル

README ファイルには、製品資料に対する変更と訂正も含めて、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、インストール CD の /opt/ibmcmp/vacpp/8.0/ ディレクトリーおよびルート・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++ Advanced Edition V8.0 for Linux 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。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++ Advanced Edition V8.0 for Linux 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 上の次のサイトでも表示可能です。

  publib.boulder.ibm.com/infocenter/lnxpcomp/index.jsp

- PDF 文書

  PDF 文書は、デフォルトでは、/opt/ibmcmp/vacpp/8.0/doc/*language*/PDF/ ディレクトリーにあり、また Web 上の次のサイトでも入手可能です。

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

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

表 2. 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++ Advanced Edition V8.0 for Linux インストール・ガイド*, GD88-6731-00」	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++ Advanced Edition V8.0 for Linux コンパイラー解説書*, SD88-6729-00」	compiler.pdf	並列処理に使用されるものも含めて、種々のコンパイラー・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に関する情報が記載されています。
「*IBM XL C/C++ Advanced Edition V8.0 for Linux ランゲージ・リファレンス*, SC88-9978-01」	language.pdf	移植性および非専有規格への準拠に関する言語の拡張機能も含めて、IBM によってサポートされる C および C++ プログラム言語に関する情報が記載されています。

表 2. XL C/C++ PDF ファイル (続き)

資料の表題	PDF ファイル名	説明
「*IBM XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*, SC88-9976-01」	proguide.pdf	アプリケーションの移植、Fortran コードによる言語間呼び出し、ライブラリー開発、アプリケーションの最適化と並列処理、および XL C/C++ 高性能ライブラリーといった高度なプログラミング上のトピックに関する情報が記載されています。

これらの PDF ファイルは、Adobe Reader から表示可能で印刷可能です。 Adobe Reader をインストールしていない場合には、次のサイトからダウンロードすることができます。

www.adobe.com

追加資料

Redbooks、公式報告書、チュートリアル、およびその他の論文を含めて、XL C/C++ に関連する上記以外の資料が、Web 上の次のサイトで使用可能です。

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

技術サポート

XL C/C++ Support ページから追加の技術サポートを入手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サポート FAQ および他のサポート資料に対する検索機能を備えたポータルを提供します。 Web 上の次のサイトで XL C/C++ Support ページを見つけることができます。

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

必要なものが見つからない場合には、次のアドレスに E メールを出して問い合わせることができます。

compinfo@ca.ibm.com

XL C/C++ に関する最新の情報に関しては、次の場所にある製品情報サイトを参照してください。

www.ibm.com/software/awdtools/xlcpp

第 1 章 XL C/C++ の機能の概要

XL C/C++ Advanced Edition V8.0 for Linux は、Fortran プログラムとの間での言語間呼び出しを含めて、大規模な、複合の、コンピューターを集約的に使用するプログラム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、XL C/C++ を評価する人、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 XL 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。

XL コンパイラーは、AIX、Linux 配布版、OS/390、OS/400、z/OS、および z/VM オペレーティング・システムといった多様なプラットフォームおよびプログラム言語間でコンパイラーの関数および最適化テクノロジーを共用する共通のコード・ベースから導き出された、より大きい IBM C、C++、および Fortran コンパイラーのファミリーの一部です。この共通のコード・ベースは、国際的プログラム言語標準への準拠と共に、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確実にするのに役立ちます。

XL コンパイラーは、AIX および選択された Linux 配布版で使用することができます。

資料、オンライン・ヘルプ、および技術サポート

このガイドは、XL C/C++ およびその機能の概要を記載しています。ユーザーは、以下の形式での更に広範囲な製品資料を見つけることができます。

- README ファイル。
- インストール可能 man ページ。
- HTML ベースの情報システム。
- Portable Document Format (PDF) 文書。
- Web 上のオンライン技術サポート。

XL C/C++ で提供される製品資料および技術サポートに関する詳細については、以下の項を参照してください。

- viii ページの『IBM XL C/C++ の資料』
- x ページの『追加資料』
- x ページの『技術サポート』

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++ Advanced Edition V8.0 for Linux は、いくつかの Linux 配布版 をサポートします。サポートされる配布版および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++ Advanced Edition V8.0 for Linux インストール・ガイド*」 の中の 『システムの前提条件』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とディスク・スペースを備えたすべての POWER3™、POWER4™、 POWER5™、POWER5+™、PowerPC®、および PowerPC 970 システムで実行します。

さまざまなハードウェア構成から最大の利益を得るために、コンパイラーは、アプリケーションの実行に使用されるマシンの構成に基づいて、パフォーマンス・チューニング用のいくつかのオプションを提供しています。

高度に構成すること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++ は、コンパイラーを呼び出すのに使用できるいくつかの互いに相異なるコマンドを提供しています。たとえば、**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ようにコンパイル出力を調整するように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 および C++ 言語レベル、および多くの使用頻度の高い拡張機能をサポートするように提供されています。

コンパイラーは、また、大部分の呼び出しコマンドの、対応する「**_r**」バージョンも提供しています。たとえば、**xlC_r** です。これらの「**_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バインドするように指示し、コンパイラーが作成するデータおよびプロシージャー用のスレッド・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関する詳細については、本書の 24 ページの『XL C/C++ によるコンパイル』または「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイラーまたはコンパイラー・コンポーネントの呼び出し』を参照してください。

コンパイラー・オプション

提供されている大きいコンパイラー・オプションのセットを使用して、コンパイラーのアクションを制御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および調整し、他のプラットフォームからのプログラム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++ 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 C または C++ 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・オプションを指定させます。

XL C/C++ コンパイラー・オプションの詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイラー・オプションの参照』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、/etc/opt/ibmcmp/vac/8.0/vac.cfg にデフォルトのコンパイラー構成ファイルを作成します。この構成ファイルには、コンパイラー・オプションのデフォルトの設定値を定義するいくつかのスタンザが入っています。

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。そのような場合、XL C/C++ は、追加の構成ファイルを作成するのに使用できる **vac_configure** ユーティリティーを提供します。 次に、ユーザーは、任意のテキスト・エディターを使って、これらのファイルに独自の頻繁に使用するコンパイラー・オプションの設定値を入れるように変更することができます。

カスタム構成ファイルの作成と使用に関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『構成ファイルのカスタマイズ』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 および 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のほかに、XL C/C++ は、以下のものが含まれる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 拡張機能
- VMX 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

C および C++ の言語仕様および拡張機能の詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux 言語解説書*」の中の『サポートされる言語標準』を参照してください。

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。コンパイラーは、コンパイラーを呼び出す前に、これらのオプションを、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。

gxlc および **gxlc++** 呼び出しコマンドは、/etc/opt/ibmcmp/vac/8.0/gxlc.cfg プレーン・テキスト構成ファイルを使用して、GNU 対 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、/etc/opt/ibmcmp/vac/8.0/gxlc.cfg ファイルをカスタマイズして、ユーザーがもっている可能性のある任意の固有のコンパイル要件をよりよく満たすことができます。詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」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 コンパイラー、GCC バージョン 3.3 を使用して作成されたコードとバイナリー互換のコードを作成します。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。

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 C および 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。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のメッセージを出します。

詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『**-qlanglvl**』を参照してください。

ライブラリー

XL C/C++ Advanced Edition V8.0 for Linux は、以下のライブラリーとともに出荷されます。

- SMP Runtime ライブラリーは、明示的および自動化された、両方の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、調整された数学の組み込み関数の Mathematics Acceleration Subsystem (MASS) ライブラリー
- 調整された代数関数の Basic Linear Algebra Subsystem (BLAS) ライブラリー
- C++ C++ Runtime ライブラリーには、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。

Mathematics Acceleration Subsystem ライブラリー

IBM Mathematics Acceleration Subsystem (MASS) ライブラリーは、特に PowerPC 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、高度に調整されたスカラーおよびベクトルの数学的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・ファミリー用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、それらに対応する `libm` ルーチンを超えて改良されたパフォーマンスを提供します。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 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++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『Mathematical Acceleration Subsystem の使用』を参照してください。

Basic Linear Algebra Subprogram

高性能代数関数の 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加算を実行します。

BLAS 関数の使用に関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『Basic Linear Algebra Subprogram 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

xlc_install

この対話式ユーティリティーは、システムに XL C/C++ をインストールするのに役立ちます。

new_install

XL C/C++ 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

PDFDIR ディレクトリーの管理に使用される、プロファイル指示フィードバックに関連するコマンド。指定されたディレクトリー、PDFDIR ディレクトリー、または現行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

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する、プロファイル指示フィードバック (PDF) に関連するコマンド。 PDF レコードは、同じ実行可能モジュールから得られたものである必要があります。

resetpdf コマンド

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

showpdf コマンド

プロファイル指示フィードバック・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するコマンド。

gxlc および gxlc++ ユーティリティー

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、XL C/C++ コンパイラーを呼び出す呼び出しメソッド。これらのユーティリティーの目的は、GNU コンパイラーで作成された既存のアプリケーション用に使用される makefile への変更の数を最小化し、XL C/C++ 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実行時の全体的パフォーマンスを向上させることができます。 C および 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・コードを再構成して、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コードを作成できるようになります。

可能であれば、コードを最適化する前に、まず最適化しない状態でコードをテストおよびデバッグしてください。

最適化手法の詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『アプリケーションの最適化』を参照してください。

最適化関連のコンパイラー・オプションの要約については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『パフォーマンス最適化オプション』を参照してください。

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い実行可能モジュールをサポートするために、64 ビット実行可能モジュールの要件を満たす、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 64 ビット・オブジェクトをバインドして 64 ビット実行可能モジュールを作成します。一つ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ることに注意してください。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
- 32 ビット・プラットフォーム上で 64 ビット・アプリケーションを実行しようとする試み

32 ビット実行可能モジュールは、現在 32 ビット・プラットフォームで実行しているのと同じように、64 ビット・プラットフォームおよび 32 ビット・プラットフォームの両方で、依然として実行し続けます。

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++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『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++ Advanced Edition V8.0 for Linux 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 (OpenMP)
- 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)
- POSIX スレッド (Pthreads) 並列処理
- `fork()` および `exec()` を使用した下位 UNIX 並列処理

詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」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 のコンパイラーによってサポートされるされる 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、www.openmp.org を参照してください。

診断リスト

コンパイラー出力リストには、組み込んでも、あるいは省略してもいい任意指定の部分があります。適用できるコンパイラー・オプションおよびリスト自体に関する詳細については、34 ページの『XL C/C++ コンパイラー・リスト』を参照してください。

シンボリック・デバッガー・サポート

ユーザーのプログラムをデバッグするときには、 **gdb** または他の任意のシンボリック・デバッガーを使用することができます。

第 2 章 V8.0 の新機能

XL C/C++ Advanced Edition V8.0 for Linux の新規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。

- 『パフォーマンスおよび最適化』
- 16 ページの『言語の機能拡張および API に対するサポート』
- 16 ページの『使いやすさ』
- 17 ページの『新規のコンパイラー・オプション』

パフォーマンスおよび最適化

多数の新機能と機能拡張は、最適化とパフォーマンス・チューニングのカテゴリーに分類されます。

アーキテクチャーおよびプロセッサー固有のコードのチューニング

-qarch コンパイラー・オプションは、指定されたマシン・アーキテクチャー用に生成される特定の命令を制御します。 **-qtune** コンパイラー・オプションは、命令、スケジューリング、および他の最適化を調整して、指定されたハードウェアでのパフォーマンスを向上させます。これらのオプションは一緒に働いて、指定されたアーキテクチャーに最適なパフォーマンスを与えるアプリケーション・コードを生成します。

XL C/C++ V8.0 は、VMX 命令セットをサポートするプロセッサーおよび新しく使用可能になった POWER5+ プロセッサーをサポートする **-qarch** コンパイラー・オプションで使用可能なサブオプションのリストを増加させます。以下の新規の **-qarch** オプションが使用可能です。

- -qarch=ppc64v
- -qarch=pwr5x

高性能ライブラリー

XL C/C++ には、数学集中アプリケーションのパフォーマンスを大幅に向上させることができる高度に調整された数学関数が組み込まれています。これらの関数は、以下の高性能ライブラリーを通じて提供されます。

Mathematical Acceleration Subsystem (MASS)

MASS ライブラリーは、共通の数学的計算を実行するために、高性能のスカラーおよびベクトル関数を提供しています。 XL C/C++ Advanced Edition V8.0 for Linux で組み込まれている MASS ライブラリーは、新規のスカラーおよびベクトルの関数、および POWER5 プロセッサー・アーキテクチャーの新規のサポートを導入しています。

MASS ライブラリーの使用に関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『Mathematical Acceleration Subsystem の使用』を参照してください。

Basic Linear Algebra Subprograms (BLAS)

XL C/C++ Advanced Edition V8.0 for Linux は、高性能代数関数の BLAS セットを導入しています。これらの関数を使用して、以下のことを行うことができます。

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

BLAS 関数の使用に関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド*」の中の『Basic Linear Algebra Subprogram の使用』を参照してください。

他のパフォーマンス関連コンパイラー・オプションおよびディレクティブ

次の表の項目は、前掲のセクションではまだ言及されていな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。これらのコンパイラー・オプションに関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『パフォーマンス最適化オプション』を参照してください。

表 3. 他のパフォーマンス関連コンパイラー・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
-qfloat	**-qfloat** は、以下のサブオプションを追加しています。 **-qfloat=relax** このサブオプションは、より大きい速度との交換で、一般的には右側のゼロに関係する加算および減算といった平凡な浮動小数点算術演算を除去することによって、厳密な IEEE への準拠を緩めます。 **-qfloat=norelax** これはデフォルトです。厳密な IEEE への準拠が保たれます。

表 3. 他のパフォーマンス関連コンパイラー・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
-qipa	**-qipa** は、以下のサブオプションを追加しています。 **-qipa=clonearch=***arch*{**,***arch*} 同じ命令セットの複数のバージョンが作成される 1 つ以上のプロセッサー・アーキテクチャーを指定します。 XL C/C++ は、命令セットが生成されることになる複数の特定のプロセッサー・アーキテクチャーをユーザーに指定させます。実行時に、アプリケーションは、稼働環境の特定のアーキテクチャーを検出し、そのアーキテクチャー用に専門化された命令セットを選択します。 **-qipa=cloneproc=***name*{**,***name*} **clonearch** サブオプションによって指定されたプロセッサー・アーキテクチャーについて複製するための 1 つ以上の関数の名前を指定します。
-O	**-O3** コンパイラー・オプションを指定することは、**-qhot=level=0** コンパイラー・オプションの設定も想定するようにコンパイラーに指示するようになりました。 **-O4** または **-O5** コンパイラー・オプションを指定することは、**-qhot=level=1** コンパイラー・オプションの設定も想定するようにコンパイラーに指示するようになりました。

このリリースでの新規組み込み関数

次の表は、このリリースでの新規の組み込み関数をリストしています。 XL C/C++ が提供する組み込み関数の詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『POWER および PowerPC アーキテクチャーの組み込み関数』を参照してください。

表 4. XL C/C++ 用の組み込み関数

関数	説明
`void __builtin_return_address (unsigned int level);`	現行の関数の、あるいはその 1 つの呼び出し元の戻りアドレスを戻します。ここで、*level* は、呼び出しスタックをスキャンアップするフレームの数を示す定数リテラルです。
`void __builtin_frame_address (unsigned int level);`	現行の関数の関数フレームの、またはその 1 つの呼び出し元のアドレスを戻します。ここで、*level* は、呼び出しスタックをスキャンアップするフレームの数を示す定数リテラルです。
`int __compare_and_swap(volatile int* addr, int* old_val_addr, int new_val);`	単一ワード変数の内容を、保管された古い値と比較するアトミック操作を実行します。

表 4. XL C/C++ 用の組み込み関数 (続き)

関数	説明
int __compare_and_swaplp(volatile long* *addr*, long* *old_val_addr*, long *new_val*);	ダブルワード変数の内容を、保管された古い値と比較するアトミック操作を実行します。
int __fetch_and_add(volatile int* *addr*, int *val*);	*addr* によって指定された単一ワードを、単一のアトミック操作で *val* によって指定された量だけ増量します。
long __fetch_and_addlp(volatile long* *addr*, long *val*);	*addr* によって指定されたダブルワードを、単一のアトミック操作で *val* によって指定された量だけ増量します。
unsigned int __fetch_and_and(volatile unsigned int* *addr*, unsigned int *val*);	*addr* によって指定された単一ワード内のビットを、単一のアトミック操作でその値を入力 *val* パラメーターと AND することによって、クリアします。
unsigned long __fetch_and_andlp(volatile unsigned long* *addr*, unsigned long *val*);	*addr* によって指定されたダブルワード内のビットを、単一のアトミック操作でその値を入力 *val* パラメーターと AND することによって、クリアします。
unsigned int __fetch_and_or(volatile unsigned int* *addr*, unsigned int *val*);	*addr* によって指定された単一ワード内のビットを、単一のアトミック操作でその値を入力 *val* パラメーターと OR することによって、設定します。
unsigned long __fetch_and_orlp(volatile unsigned long* *addr*, unsigned long *val*);	*addr* によって指定されたダブルワード内のビットを、単一のアトミック操作でその値を入力 *val* パラメーターと OR することによって、設定します。
unsigned int __fetch_and_swap(volatile unsigned int* *addr*, unsigned int *val*);	*addr* によって指定された単一ワードを、単一のアトミック操作で値または入力 *val* パラメーターに設定して、メモリー・ロケーションの元の内容を戻します。
double __frim(double *val*);	double 形式の入力 *val* を取り込み、*val* を次に低い整数値に丸め、結果を double 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
float __frims(float *val*);	float 形式の入力 *val* を取り込み、*val* を次に低い整数値に丸め、結果を float 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
double __frin(double *val*);	double 形式の入力 *val* を取り込み、*val* を最も近い整数値に丸め、結果を double 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
float __frins(float *val*);	float 形式の入力 *val* を取り込み、*val* を最も近い整数値に丸め、結果を float 形式で戻します。 POWER5+ プロセッサーでのみ有効です。

表 4. XL C/C++ 用の組み込み関数 (続き)

関数	説明
`double __frip(double val);`	double 形式の入力 *val* を取り込み、*val* を次に高い整数値に丸め、結果を double 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
`float __frips(float val);`	float 形式の入力 *val* を取り込み、*val* を次に高い整数値に丸め、結果を float 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
`double __friz(double val);`	double 形式の入力 *val* を取り込み、*val* をゼロに最も近い次の整数値に丸め、結果を double 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
`float __frizs(float val);`	float 形式の入力 *val* を取り込み、*val* をゼロに最も近い次の整数値に丸め、結果を float 形式で戻します。 POWER5+ プロセッサーでのみ有効です。
`long __ldarx(volatile long* addr);`	Load Double Word And Reserve Indexed (ldarx) 命令を生成します。この命令は、後続の stwcx. 命令と一緒に使用して、指定されたメモリー・ロケーションに対して読み取り/変更/書き込みをインプリメントすることができます。
`int __lwarx(volatile int* addr);`	Load Word And Reserve Indexed (lwarx) 命令を生成します。この命令は、後続の stwcx. 命令と一緒に使用して、指定されたメモリー・ロケーションに対して読み取り/変更/書き込みをインプリメントすることができます。
`int __stdcx(volatile long* addr, long val);`	Store Double Word Conditional Indexed (stdcx.) 命令を生成します。この命令は、先行の ldarx. 命令と一緒に使用して、指定されたメモリー・ロケーションに対して読み取り/変更/書き込みをインプリメントすることができます。
`int __stwcx(volatile int* addr, int val);`	Store Word Conditional Indexed (stwcx.) 命令を生成します。この命令は、先行の lwarx. 命令と一緒に使用して、指定されたメモリー・ロケーションに対して読み取り/変更/書き込みをインプリメントすることができます。
`unsigned long __mftb();`	Move From Time Base (mftb) ハードウェア命令を生成します。
`unsigned int __mftbu();`	Move From Time Base Upper (mftbu) ハードウェア命令を生成します。

言語の機能拡張および API に対するサポート

API および言語の機能拡張は、ユーザーがご使用のハードウェア・プラットフォームの能力をさらに十分に活用するコードを開発するのを容易にすると同時に、アプリケーションの開発時に、より高度な使いやすさと柔軟性を提供することができます。

C、C++、および Fortran に対する OpenMP API V2.5 サポート

XL C/C++ は OpenMP API V2.5 規格をサポートするようになりました。この最新レベルの OpenMP 仕様は、前の C/C++ および Fortran OpenMP 仕様を C/C++ と Fortran の両方のための単一仕様にまとめ、以前それらの間に存在していた不整合を解決しています。

OpenMP アプリケーション・プログラム・インターフェース (API) は、C、C++、および Fortran アプリケーションにおいて、ユーザー指示共用メモリー並列処理を開発する標準インターフェースを提供する、移植可能で拡張が容易なプログラミング・モデルです。この仕様は、OpenMP 組織という、IBM を含むコンピューター・ハードウェアおよびソフトウェア・ベンダーのグループによって定義されています。

OpenMP 仕様に関する詳細な情報は、次のサイトで見つけることができます。

www.openmp.org

使いやすさ

XL C/C++ には、ユーザーが自身のアプリケーション開発のために、コンパイラーをより容易に使用できるよう支援する、以下の新機能が組み込まれています。

新規のインストールおよび構成ユーティリティー

このリリースの XL C/C++ は、ご使用のシステムでの最初の使用に備えて、コンパイラーを簡単にインストールし構成するのを助けるために、 **xlc_install** および **new_install** ユーティリティーを導入しています。

IBM Tivoli License Manager のサポート

IBM Tivoli License Manager (ITLM) は、ユーザーがサポートされるシステム上でのソフトウェア使用量の計量およびライセンス割り振りサービスを管理するのを支援できる、Web ベースのソリューションです。一般に、ITLM は、ご使用のシステム上にインストール済みで使用中の製品を認識し、モニターします。

IBM XL C/C++ Advanced Edition V8.0 for Linux は、ITLM が、インベントリーと使用のシグニチャー・サポートに対応しています。これは、ITLM が、XL C/C++ の製品がインストールされていること、およびそれが使用されていることの両方を検出できることを意味しています。

注: ITLM は、XL C/C++ コンパイラー・オファリングの一部ではないので、別個に購入しインストールする必要があります。

いったんインストールされ活動化されると、ITLM は、ご使用のシステムを対象に、ある特定の製品がご使用のシステムにインストールされているかどうかを示す製品インベントリー・シグニチャーをスキャンします。 ITLM は、また、該当の製品のバージョン、リリース、およびモディフィケーション・レベルを識別します。XL C/C++ のシグニチャー・ファイルは、以下のディレクトリーにインストールされています。

デフォルト・インストール

`/opt/ibmcmp/vac/8.0`

デフォルトでないインストール

compiler`/vac/8.0`。ここで *compiler* は、**--prefix** インストール・オプションで指定されたインストールのターゲット・ディレクトリーです。

IBM Tivoli License Manager Web に関する詳細については、

www.ibm.com/software/tivoli/products/license-mgr

を参照してください。

新規のコンパイラー・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。

新規のコマンド行オプション

次の表は、XL C/C++ で新しいコマンド行オプションを要約したものです。すべてのコンパイラー・オプションの詳細な構文上および使用上の情報は、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイラー・オプションの解説』で見つけることができます。

オプション	説明および注釈
-qasm	**-qasm** コンパイラー・オプションは、新規の機能を追加しています。ユーザーは、このコンパイラー・オプションを使用してユーザーのプログラム内のインラインのアセンブラー・ステートメントがどのように解釈されるかを制御することができるようになったばかりでなく、asm 文に対してコードが出されるかどうかを制御することもできます。
-qasm_as	**-qasm_as** コンパイラー・オプションの構文はわずかに変わりました。
-qdump_class_hierarchy	このオプションが有効になっている場合、コンパイラーは、ファイルに対する各クラス・オブジェクトの階層および仮想関数テーブルのレイアウトの表現をダンプします。
-qlist	**-qlist** コンパイラー・オプションは、新規の **offset** および **nooffset** のサブオプションを追加しています。 **-qlist=offset** を指定すると、コンパイラーは、コード生成の始まりからではなく、プロシージャーの始まりからの、オブジェクト・リスト・オフセットを表示するように指示されます。

オプション	説明および注釈
-qmakedep	**-qmakedep** コンパイラー・オプションは、新規の **gcc** サブオプションを追加しています。 **-qmakedep=gcc** を指定すると、コンパイラーは、GNU C/C++ コンパイラーが使用するのと同様な形式で依存関係作成情報を生成するように指示されます。
-MF	この新規のコンパイラー・オプションは、**-qmakedep** または **-M** オプションによって生成された依存関係作成ファイルのファイル名を指定します。
-qppline	この新規のコンパイラー・オプションは、プリプロセスされる出力内に #line ディレクティブを生成することを可能にします。 **-qnoppline** コンパイラー・オプションは、#line ディレクティブの生成を使用不可にします。
-qreserved_reg	この新規のコンパイラー・オプションは、1 つ以上のレジスター名を予約させます。予約されたレジスターは、スタック・ポインターまたはフレーム・ポインターとして、あるいはその他の決まった役割で使用される場合を除き、コンパイル中は使用できません。
-qsourcetype	このリリースは、**-qsourcetype** コンパイラー・オプションに、新規のサブオプションとして **assembler-with-cpp** を追加しています。 通常、コンパイラーは、プリプロセッシングが必要なアセンブラーのソース・ファイルを、ファイルの **.S** ファイル名サフィックスによって認識します。コンパイラーは、**.S** ソース・ファイルをプリプロセスし、次にプリプロセッサーの出力をアセンブラーに送ります。 コマンド行上に **-qsourcetype=assembler-with-cpp** *filename* を指定すると、コンパイラーは、**assembler-with-cpp** より後に現れるすべてのファイル名を、ファイル名サフィックスには関係なく、プリプロセッシングが必要なアセンブラー・ソース・ファイルとして扱うように指示されます。
-qtmplinst	この新規のコンパイラー・オプションは、コンパイラーがテンプレートの暗黙的インスタンス化を実行する方法を管理します。
-qversion	**-qversion** コンパイラー・オプションを指定すると、正式なコンパイラー製品名およびバージョンが戻されます。

新規のプラグマ・ディレクティブ

次の表は、XL C/C++ での新規のプラグマ・ディレクティブ・オプションを要約したものです。詳細な構文および使用の情報は、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『XL C/C++ プラグマ』で見つけることができます。

#pragma ディレクティブ	説明および注釈
altivec_vrsave	altivec_vrsave ディレクティブが有効になっている場合、関数のプロローグおよびエピローグには、VRSAVE レジスターを維持するコードが組み込まれています。このプラグマは、**-qaltivec** が有効になっているときに限り効果をもち、ある 1 つの関数内でのみ使用しなければならず、それが現れる関数のみに影響を与えます。
STDC CX_LIMITED_RANGE	STDC CX_LIMITED_RANGE ディレクティブは、そのスコープ内で、中間の計算がオーバーフローを起こしたり有効桁を失ったりしないような値でのみ複素数の部分と絶対値が呼び出され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

この章では、特定のセットアップとカスタマイズのトピックを詳細に説明する他の資料を指す参照用の情報とともに、XL C/C++ のセットアップとカスタマイズに関する概要の情報を記載します。

環境変数と XL C/C++

XL C/C++ は、いくつかの環境変数を使用して、コンパイラーの操作のさまざまな局面を制御します。環境変数は、以下に示す 2 つの基本的カテゴリーに分類されます。

- コンパイラーの基本的作動環境を定義する環境変数
- ランタイム・コンパイラー・オプション・デフォルトを定義する環境変数

コンパイラー作動環境の設定

以下の環境変数は、ご自身が選択される各国語の指定、あるいはライブラリーまたは一時ファイルのロケーションの定義も含めて、コンパイラーの基本的作動環境を定義します。完全な情報については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイル環境のセットアップ』を参照してください。

LANG 診断メッセージおよびコンパイラー・リストを表示するのに使用されるデフォルトの各国語*ロケール*を指定します。この環境変数は、ランタイム動作に影響を与えます。

MANPATH
システム、コンパイラー、およびサード・パーティーの man ページ用の検索パスを指定します。

NLSPATH
メッセージ・カタログを見つけることができる 1 つ以上のディレクトリーのロケーションを指定します。この環境変数は、ランタイム動作に影響を与えます。

PDFDIR
-qpdf オプションを使用してコンパイルする場合に、プロファイル指示フィードバック情報が保管されるディレクトリーのロケーションを指定します。

TMPDIR
プログラムのコンパイル時に作成される一時ファイルをコンパイラーが保管するディレクトリーのロケーションを指定します。この環境変数は、ランタイム動作に影響を与えます。

デフォルトのランタイム・オプションの設定

以下の環境変数は、コマンド行で、またはご自身のプログラム・ソース内にあるディレクティブ内で指定されるコンパイラー・オプションの設定値によって、明示的にオーバーライドされない限り、コンパイラーによって使用されるランタイム・コ

ンパイラー・オプション・デフォルトを定義します。完全な情報については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイル環境のセットアップ』を参照してください。

XL_NOCLONEARCH

汎用コードが、特定のプロセッサー・アーキテクチャー用にバージョン処理されていないコンパイル済みオブジェクト・コードである汎用コードのみを実行するようにプログラムに指示します。 XL_NOCLONEARCH 環境変数を設定して、ユーザーが自身のアプリケーションをデバッグするのを助けることができます。

XLSMPOPTS

XLSMPOPTS 環境変数を使用すると、SMP の実行に影響を与えるランタイム・オプションを指定することができます。

OMP_DYNAMIC、OMP_NESTED、OMP_NUM_THREADS、OMP_SCHEDULE

これらの環境変数は、OpenMP 規格の一部です。これらの環境変数を使用すると、アプリケーションが並列コードのセクションを実行する方法を指定することができます。

構成ファイルのカスタマイズ

構成ファイルは、コンパイラー・オプションおよびコンパイラー呼び出し用のデフォルト設定値を指定するプレーン・テキスト・ファイルです。 XL C/C++ は、コンパイラーのインストール時に、ファイル /etc/opt/ibmcmp/vac/8.0/vac.cfg で、デフォルト構成を提供します。

ユーザーが単一ユーザー・システム上で実行している場合、またはコンパイル・スクリプトまたは Make ファイルを備えたコンパイル環境をすでにもっている場合には、デフォルト構成ファイルをそのままにしておくことが必要です。

代わりに、特定のアプリケーションまたはアプリケーション・グループによって要求される特別なコンパイル要件を満たすために、追加のカスタム構成ファイルを作成することもできます。

カスタム構成ファイルの作成と使用に関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『構成ファイルのカスタマイズ』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

特定のマシンにどのレベルの XL C/C++ がインストールされているかがわからない場合があります。ソフトウェア・サポートに連絡を取る場合にこの情報を知る必要が生じます。

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 PTF リリース・レベルを表示するために、**『-qversion』** コンパイラー・オプションを使用してコンパイラーを呼び出します。たとえば次のようにします。

```
xlC -qversion
```

第 4 章 XL C/C++ でのプログラムの編集、コンパイル、およびリンク

基本的な C および C++ プログラム開発は、編集、コンパイル、およびリンク (デフォルトで、コンパイルと組み合わせられた単一ステップ)、および実行の反復されるサイクルから成り立ちます。

前提条件の情報:

1. コンパイラーを使用できるようになるためには、まず、すべての Linux 設定値 (たとえば特定の環境変数およびストレージの制限) が正しく構成されていることを確かめる必要があります。詳細については、21 ページの『環境変数と XL C/C++』を参照してください。
2. C および C++ プログラムの作成に関してさらに学習するには、「*XL C/C++ Advanced Edition V8.0 for Linux 言語解説書*」を参照してください。

コンパイラー・フェーズ

典型的なコンパイラー呼び出しコマンドは、以下に挙げるプログラムのうちいくつか、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、フェーズの中のあるものは、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. プリプロセッサー
2. 以下のフェーズから成り立つコンパイラー。
   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. ループ変換
   c. プロシージャー間分析
   d. 最適化
   e. レジスター割り振り
   f. 最終アセンブリー
3. アセンブラー (プリプロセスされた後の **.s** ファイルおよび **.S** ファイルについて)
4. リンカー **ld**

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見るためには、ユーザーが自身のアプリケーションをコンパイルするときに、**-qphsinfo** および **-v** コンパイラー・オプションを指定します。

C および C++ ソース・ファイルの編集

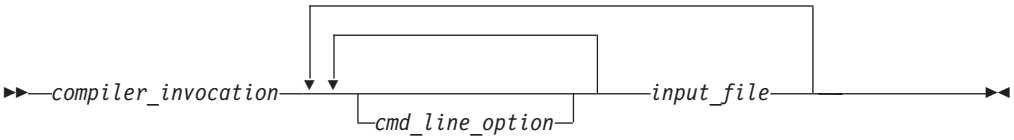
C および C++ ソース・プログラムを作成するには、**vi** または **emacs** といった、使用可能な任意のテキスト・エディターを使用することができます。構成ファイルが追加の非標準ファイル名サフィックスをまたは **-qsourcetype** コンパイラー・オプションが追加の非標準ファイル名サフィックスを定義するのでない限り、ソース・

プログラムは認識されたファイル名サフィックスを使用する必要があります。 XL C/C++ によって認識されたファイル名サフィックスのリストについては、 26 ページの『XL C/C++ 入力ファイル』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、そのソース・プログラムは、「*XL C/C++ Advanced Edition V8.0 for Linux 言語解説書*」に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

ソース・プログラムをコンパイルするには、次に示す構文をもつコンパイラー呼び出しコマンドのうちの 1 つを使用します。



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、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。

新規の C または C++ アプリケーションの作業については、xlC または xlc++、またはスレッド・セーフでそれに対応するものを使用してコンパイルすることを考慮する必要があります。

xlC と **xlc++** は両方ともプログラム・ソースを C かまたは C++ としてコンパイルしまが、C++ ファイルを **xlc** でコンパイルすると、C コンパイラーによってリンカーが呼び出されたときに C++ コードに必要なライブラリーが指定されていないために、リンクまたはランタイムのエラーが生じる結果になることがあります。他の基本コンパイラー呼び出しコマンドは、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・サポートを提供するために存在します。

基本コンパイラー呼び出しコマンドに加えて、XL C/C++ は、また、多くの基本コンパイラー呼び出しの、専門化されたバリアントを提供します。基本コンパイラー呼び出しに対するバリエーションには、その呼び出しコマンドの名前にサフィックスを付加することによって、名前が付けられます。呼び出しのバリアントのサフィックスの意味は以下のとおりです。

_r **-qsmp** あるいは IBM SMP が含まれているソース・コードを使用してコンパイルされたアプリケーションも含めたマルチスレッド・アプリケーション用の POSIX Pthread API をサポートするスレッド・セーフ呼び出しバリアント

表 5. XL C/C++ コンパイラー呼び出しコマンド

基本 呼び出し	基本呼び出しのバリアント	説明
xlC xlc++	xlC_r xlc++_r	ソース・ファイルを C++ 言語ソース・コードとしてコンパイルするために、コンパイラーを呼び出します。
xlc	xlc_r	ソース・ファイルを C ソース・コードとしてコンパイルするように、コンパイラーを呼び出します。
c99	c99_r	ISO C99 規格 (ISO/IEC 14882:1999) へ厳密に準拠してソース・ファイルがコンパイルされるようにコンパイラーを呼び出します。 **注:** ISO C99 規格は、ランタイム・ライブラリーの機能も指定します。これらの機能は、現在ご使用のシステムにインストールされているランタイム・ライブラリーではサポートできません。
c89	c89_r	ISO C89 規格 (ISO/IEC 9899:1990) へ厳密に準拠してソース・ファイルがコンパイルされるようにコンパイラーを呼び出します。
cc	cc_r	C89 または C99 への準拠を必要としないレガシー C コードを使用するように、コンパイラーを呼び出します。
gxlc		GNU C コマンド行オプションを XL C/C++ オプションへ変換した後、コンパイラーを呼び出します。 **注:** すべての GNU C オプションが完全に XL C/C++ に対応するものをもっているわけではありません。
gxlC gxlc++		GNU C++ コマンド行オプションを XL C/C++ オプションへ変換した後、コンパイラーを呼び出します。 **注:** すべての GNU C++ オプションが完全に XL C/C++ に対応するものをもっているわけではありません。

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・セーフのコンパイル呼び出しを提供します。

- xlC_r
- xlc++_r
- xlc_r
- c99_r
- c89_r
- cc_r

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

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にはなりません。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。今度は、**-qsmp** オプションの方は、これらの 6 つの呼び出しコマンドのうちの 1 つと一緒にのみ指定することができます。 **-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。

XL C/C++ 入力ファイル

コンパイラーへの入力ファイルには次のものがあります。

ソース・ファイル (C 言語用の .c サフィックス、 C++ 言語用の .C .cc .cp .cpp .cxx .c++ サフィックス)

コンパイラーは、これらのサフィックスの付いたファイルを、コンパイル用の C または C++ ソース・ファイルであると見なします。

コンパイラーは、指定されたソース・ファイルをコマンド行で指定された順序でコンパイルします。指定されたソース・ファイルが見つからない場合、コンパイラーはエラー・メッセージを作成し、次のファイルがあれば、そのファイルに進みます。

標準の C または C++ ファイル命名規則に準拠しない C または C++ ソース・ファイルをもっている場合には、**-+** コンパイラー・オプションを使用して、コンパイラーに、そのようなファイルを C または C++ ソース・ファイルとして扱うように指示することができます。そのようなファイルは、**-+** コンパイラー・オプションが有効である場合、.a、.o、.so、.s、または .S ファイル名サフィックスをもつものを除き、C++ ソース・ファイルとしてコンパイルされます。

インクルード・ファイルには、ソースも含まれており、しばしば、通常 C または C++ ソース・ファイル用に使用されるものとは異なるサフィックスをもちます。

プリプロセスされるソース・ファイル (.i サフィックス)

コンパイラーは、プリプロセスされるソース・ファイル *filename*.i をコンパイラーへ送り、そのファイルはそこで再度 .c または .C ファイルと同様にプリプロセスされます。プリプロセスされるファイルは、マクロおよびプリプロセッサー・ディレクティブを検査するのに有用です。

オブジェクト・ファイル (.o サフィックス)

コンパイラーは、ソース・ファイルをコンパイルした後、**ld** コマンドを使用して、結果の .o ファイル、入力ファイルとして指定した任意の .o ファイル、および製品およびシステムのライブラリー・ディレクトリー内のいくつかの .o および .a ファイル をリンクします。コンパイラーは、次に、それらのオブジェクト・ファイルから、単一の .o オブジェクト・ファイルまたは単一の実行可能出力ファイルを作成することができます。

アセンブラー・ソース・ファイル (.s および .S サフィックス)

コンパイラーは、アセンブラー・ソース・ファイルをアセンブラー (**as**) へ送ります。アセンブラーは、リンク時にオブジェクト・ファイルをリンカーへ送ります。

注: .S ファイル名サフィックスをもつアセンブラー・ソース・ファイルがコンパイラーによって最初にプリプロセスされ、次にアセンブラーへ送られます。

共用オブジェクトまたはライブラリー・ファイル (.so サフィックス)

これらは、実行時にマルチプロセスによってロードされ共用されることが可能なオブジェクト・ファイルです。リンク時に共用オブジェクトが指定されると、オブジェクトに関する情報は出力ファイルに記録されますが、共用オブジェクトからのコードは実際に出力ファイルには組み込まれません。

構成ファイル (.cfg サフィックス)

構成ファイルの内容は、コンパイル・プロセスの多くの面 (最も一般的には、コンパイラーのデフォルト・オプション) を決定します。それを使用して、各種のデフォルト・コンパイラー・オプションのセットを集中させたり、1 つのシステム上に複数のレベルの XL C/C++ コンパイラーを存在し続けさせたりすることができます。

デフォルト構成ファイルは /etc/opt/ibmcmp/vac/8.0/vac.cfg および /etc/opt/ibmcmp/vac/8.0/gxlc.cfg です。

プロファイル・データ・ファイル

-qpdf1 オプションは、後続のコンパイルで使用されるランタイム・プロファイル情報を作成します。この情報は、パターン .*pdf* に一致する名前をもった 1 つ以上の隠しファイルに保管されます。

XL C/C++ 出力ファイル

C および C++ が作成する出力ファイルは以下のとおりです。

実行可能ファイル: a.out

デフォルトで、XL C/C++ は、**a.out** という名前の実行可能ファイルを現行ディレクトリーに作成します。

オブジェクト・ファイル: *filename***.o**

-c コンパイラー・オプションを指定すると、コンパイラーは実行可能ファイルを作成する代わりに、指定された各プログラム・ソース入力ファイルに対してオブジェクト・ファイルを 1 つ作成し、アセンブラーは指定された各アセンブラー入力ファイルに対してオブジェクト・ファイルを 1 つ作成します。デフォルトで、出力オブジェクト・ファイルは、それらに対応するソース入力ファイルと同じファイル名プレフィックスを使用して、現行ディレクトリーに保管されます。

アセンブラー・ソース・ファイル: *filename***.s**

-S コンパイラー・オプションを指定すると、XL C/C++ コンパイラーは実行可能ファイルを作成する代わりに、指定された各入力ソース・ファイルに対して同等のアセンブラー・ソース・ファイルを 1 つ作成します。デフォ

ルトで、出力アセンブラー・ソース・ファイルは、それらに対応するソース入力ファイルと同じファイル名プレフィックスを使用して、現行ディレクトリーに保管されます。

コンパイラー・リスト・ファイル: *filename*.**lst**
デフォルトでは、1 つ以上のリスト関連のコンパイラー・オプションを指定しない限り、リストは作成されません。リスト・ファイルは、ソース・ファイルと同じファイル名プレフィックスを付けて現行ディレクトリーに入れられます。

cpp にプリプロセスされるソース・ファイル: *filename*.**i**
プリプロセスされるソース・ファイルを作成するには、コンパイル時に、**-P** オプションを指定します。ソース・ファイルはプリプロセスされますが、コンパイルはされません。 **-E** オプションからの出力をリダイレクトして、#line ディレクティブが含まれているプリプロセスされたファイルを生成することもできます。プリプロセスされるソース・ファイル、*filename*.i は、各ソース・ファイルごとに 1 つ作成されます。デフォルトで、出力プリプロセッサー・ソース・ファイルは、それらに対応するソース入力ファイルと同じファイル名プレフィックスを使用して、現行ディレクトリーに保管されます。

依存関係ファイルの作成: *filename*.**u**
-M または **-qmakedep** コンパイラー・オプションが有効になっているときに、コンパイラーは、コンパイルされた各 C または C++ ソース・ファイルごとに、.u ファイルを作成します。ユーザーは、.u ファイルが提供する依存関係情報を使用して、Make ファイルを作成するのに役立てることができます。

各 .u ファイルには、次のような一般的形で、入力ファイル用に 1 行、および各インクルード・ファイル用に 1 つの項目が入っています。

```
file_name.o :file_name.c
file_name.o :include_file_name
```

インクルード・ファイルは、#include プリプロセッサー・ディレクティブ用の、コンパイラー検索順序規則に従ってリストされます。見つからない場合は、インクルード・ファイルは .u ファイルに追加されません。 include 文のないファイルは、入力ファイル名だけをリストする 1 行が入っている出力ファイルを作成します。

プロファイル・データ・ファイル (.*pdf*)
これらは、**-qpdf1** コンパイラー・オプションが作成するプロファイル指示フィードバック・ファイルです。これらは、以降のコンパイルで、実際の実行結果に従って最適化を調整するために使用されます。

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機能の実行など、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。

- コマンド行で、コマンド行コンパイラー・オプションを使用して

- コンパイラー構成ファイルにあるスタンザで
- 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
- あるいは、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、コンパイラーは通常次の一般的な優先順位の順序を適用します。

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: -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。

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れらの指定の仕方に関する情報の詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『コンパイラー・オプションの参照』を参照してください。

XL C/C++ プログラムのリンク

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、自動的にリンカーを呼び出し、実行可能出力ファイルを作成します。たとえば、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。

リンクが終了したならば、31 ページの『第 5 章 XL C/C++ プログラムの実行』の指示に従ってプログラムを実行してください。

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・ファイルを該当するライブラリー
                                        にリンクする
```

コンパイラー呼び出しコマンドを介してリンカーを実行するのが最適な場合もあります。このコマンドは、いくつかの余分な **ld** オプションおよびライブラリー名をリンカーに自動的に渡すからです。

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・ルーチンとの間で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。

プログラムのリンクの詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『リンケージ・エディターの呼び出し』を参照してください。

また、ライブラリーのコンパイルとリンクに関する詳細については、「XL C/C++ Advanced Edition V8.0 for Linux プログラミング・ガイド」の中の『ライブラリーの構成』を参照してください。

第 5 章 XL C/C++ プログラムの実行

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。

プログラム実行可能ファイルの名前を、任意のランタイム引数とともにコマンド行に入力すれば、プログラムを実行できます。

間違ったコマンドを誤って実行することがあり得るので、プログラム実行可能ファイルに (たとえば **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はあり得ます。

環境変数の詳細、および環境変数がランタイムにアプリケーションにどのように影響を与えられるかに関する詳細については、21 ページの『環境変数と XL C/C++』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 インストール・イメージを、ライセンス交付および使用の情報と一緒に、次のサイトにある XL C/C++ Support ページの Download セクションから取得することができます。

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

第 6 章 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、訂正する援助として、これらのメッセージを使用することができます。

この章では、XL C/C++ が提供する主要な診断メッセージの簡潔な概要を記載します。ユーザーのアプリケーションに関する問題を解決するのを助けることができる関連コンパイラー・オプションに関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『エラー検査およびデバッグのオプション』および『リストおよびメッセージを制御するオプション』を参照してください。

コンパイルおよび戻りコード

コンパイルの終了時に、コンパイラーは、以下の任意の条件のもとで戻りコードをゼロに設定します。

- メッセージが出されていない。
- 診断されたすべてのエラーの最高重大度レベルが、**-qhalt** コンパイラー・オプションの設定値よりも小さく、しかもエラーの数が **-qmaxerr** コンパイラー・オプションで設定した限界値に達していなかった。
- **-qhaltonmsg** コンパイラー・オプションによって指定されたメッセージが出されていない。

その他の場合には、コンパイラーは戻りコードを以下のいずれかの値に設定します。

戻りコード	エラー・タイプ
1	**-qhalt** コンパイラー・オプションの設定値よりも高い重大度レベルをもつエラーが起こりました。
40	オプション・エラーまたはリカバリー不能エラーが起こりました。
41	構成ファイル・エラーが起こりました。
250	メモリー不足が起こりました。コンパイラー呼び出しコマンドは、自分が使用するメモリーをそれ以上割り振ることができません。
251	シグナル受け取りエラーが起こりました。すなわち、リカバリー不能エラーまたは割り込みシグナルが起こりました。
252	ファイル未検出エラーが起こりました。
253	入出力エラー (ファイルの読み取りおよび書き込みができない) が起こりました。
254	フォーク・エラーが起こりました。新規プロセスを作成することができません。
255	プロセスの実行中にエラーが検出されました。

注: 戻りコードは、ランタイム・エラーに対しても表示されることがあります。

XL C/C++ コンパイラー・リスト

診断情報は、**-qlist**、**-qsource**、**-qxref**、**-qattr**、**-qreport**、および **-qlistopt** コンパイラー・オプションの設定値に従って出力リストに作成されます。 **-S** オプションは、別個のファイルにアセンブラー・リストを生成します。

アプリケーションのコンパイル中にコンパイラーがプログラミング・エラーに遭遇した場合、コンパイラーは標準のエラー・デバイス、および適切なコンパイラー・オプションが選択されている場合にはリスト・ファイルに診断メッセージを出します。

リストの援助によって問題の原因を突き止めるには、以下の項目を参照することができます。

- ソース・セクション (ソース・プログラムのコンテキストでのコンパイル・エラーを見つけるため)
- 属性および相互参照セクション (名前の誤ったデータ・オブジェクト、宣言なしで使用されているデータ・オブジェクト、または一致していないパラメーターを見つけるため)
- 変換およびオブジェクトのセクション (生成されたコードが予期したとおりのものかどうかを調べるため)

見出しは、リストの主要なセクションを識別します。見出しの先頭を簡単に見つけることができるように、不等号「より大」(>) 記号のストリングがセクション見出しの前に付きます。

```
>>>>> section name
```

適切なコンパイラー・オプションを指定することによって、リストに現れるセクションを選択できます。これらのオプションに関する詳細については、「*XL C/C++ Advanced Edition V8.0 for Linux コンパイラー解説書*」の中の『エラー検査およびデバッグのオプション』および『リストおよびメッセージを制御するオプション』を参照してください。

ヘッダー・セクション

リスト・ファイルには、以下の項目が含まれているヘッダー・セクションがあります。

- 以下のものから構成されるコンパイラー ID
  - コンパイラー名
  - バージョン番号
  - リリース番号
  - モディフィケーション番号
  - 修正番号
- ソース・ファイル名
- コンパイル日付
- コンパイル時刻

ヘッダー・セクションは、リストに必ず存在します。それは最初の行であり、1 度だけ出現します。複数のコンパイル単位が存在する場合、以下のセクションは、各コンパイル単位に対して繰り返されます。

オプション・セクション

オプション・セクションは、リストに必ず存在します。コンパイル単位ごとに別個のセクションがあります。このセクションは、コンパイル単位に対して有効な、指定されているオプションを示します。この情報は、競合するオプションが指定されている場合に有用です。 **-qlistopt** コンパイラー・オプションを指定した場合、このセクションは、すべてのオプションの設定値をリストします。

ソース・セクション

ソース・セクションには、行番号とファイル番号 (任意指定) の付いた入力ソース行が含まれています。ファイル番号は、ソース行が取り出されたソース・ファイル (またはインクルード・ファイル) を示します。メインファイルのすべてのソース行 (インクルード・ファイルからのソース行ではない) には、ファイル番号は印刷されません。各インクルード・ファイルにはファイル番号が関連づけられており、インクルード・ファイルからのソース行には、そのファイル番号が印刷されます。左から、ファイル番号、その右に行番号、ソース行のテキストの順で印刷されます。コンパイラーは、各ファイルを基準として、行に番号を付けます。それらと関連づけられているソース行とソース番号は、 **-qsource** コンパイラー・オプションが有効な場合だけ印刷されます。

-qsource オプションが有効になっている場合、エラー・メッセージはソース・リスト中に混在します。コンパイル・プロセス中に生成されるエラー・メッセージには、次のものが含まれています。

- ソース行
- エラーの列を指し示す標識の行
- エラー・メッセージ

たとえば、次のとおりです。

```
      7 |     for (i=0; i<100; i++) {
      8 |         for (j=0; j<100; j++) {
      9 |             a[i:j] = j;
"loop.c", line 9.16: 1506-277 (S) Syntax error: possible missing ',' or ']'?
     10 |         }
     11 |     }
```

-qnosource オプションが有効になっている場合、エラー・メッセージはソース・セクションに表示されるものすべてであり、エラー・メッセージには以下のものが含まれます。

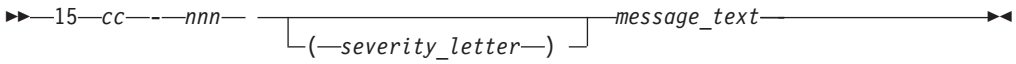
- 引用符で囲まれたファイル名
- エラーの行番号と列の位置
- エラー・メッセージ

たとえば、次のとおりです。

```
"loop.c", line 9.16: 1506-277 (S) Syntax error: possible missing ',' or ']'?
```

エラー・メッセージの形式

C および C++ 診断メッセージの形式は次のとおりです。



ここで、

15*cc nnn* XL C/C++ メッセージとコンポーネント番号を示します。

severity_letter 前のセクションで説明したように、問題がどの程度重大であるかを示します。

コンパイル・エラーは以下のような重大度レベルをもつことがあり、これはある種のエラー・メッセージの一部として表示されます。

U リカバリー不能エラー。内部コンパイラー・エラーが原因でコンパイルが失敗しました。

S 重大エラー。コンパイルが以下のいずれかの理由で失敗しました。
- コンパイラーが訂正できなかった条件が存在します。
- 内部コンパイラー・テーブルがオーバーフローしました。プログラムの処理は停止して、XL C/C++ はオブジェクト・ファイルを作成しません。
- インクルード・ファイルが存在しません。プログラムの処理は停止して、XL C/C++ はオブジェクト・ファイルを作成しません。
- リカバリー不能プログラム・エラーが検出されました。ソース・ファイルの処理は停止して、XL C/C++ はオブジェクト・ファイルを作成しません。通常このエラーは、コンパイル中に報告されたプログラム・エラーを修正して訂正することができます。

E C コンパイルのみ。コンパイラーは、ユーザーのソース・コードにエラーを検出し、それを訂正しようと試みました。コンパイラーはユーザーのアプリケーションのコンパイルを続けますが、ユーザーが期待する結果を生成しないことがあります。

W 警告メッセージ。コンパイラーは、ユーザーのソース・コードに潜在的な問題を検出しましたが、それを訂正しようとはしませんでした。コンパイラーはユーザーのアプリケーションのコンパイルを続けますが、ユーザーが期待する結果を生成しないことがあります。

I 通知メッセージ。エラーを示すわけではなく、予期しない動作を回避するために留意すべきことを示しています。

'*message text***'** エラーを説明するテキストです。

デフォルトでは、重大エラー (重大度 S) を検出すると、コンパイラーは出力ファイルを作成しないで停止します。 **-qhalt** オプションを使用して、別の重大度を指定すれば、重大度のより低いエラーに対して、コンパイラーを停止させることができます。たとえば、**-qhalt=w** を使用した場合、重大度 W またはそれ以上の重大度のエラーに遭遇するとコンパイラーが停止します。この手法を使用すると、プログラム

の構文上およびセマンティック上の妥当性をチェックするのに必要なコンパイル時間を短縮することができます。 **-qflag** オプションを使用すると、コンパイラーを停止させることなく重大度の低いメッセージを制限することができます。

変換レポート・セクション

-qreport オプションが有効になっていると、コンパイラーは、XL C/C++ がプログラムを最適化した方法を示す変換レポートを生成します。このセクションは、ユーザーが **-qhot** または **-qsmp** オプションによって生成された並列処理およびループ変換を表示することができるように、元のソース・コードに対応する疑似コードを表示します。

属性および相互参照セクション

このセクションは、コンパイル単位で使用されるエンティティーに関する情報を提供します。これが存在するのは、**-qxref** または **-qattr** コンパイラー・オプションが有効になっている場合です。有効なオプションに応じて、このセクションにはコンパイル単位で使用されるエンティティーに関する以下の情報の全部または一部が入ります。

- エンティティー名
- エンティティーの属性 (**-qattr** が有効になっている場合)。属性情報には、以下の詳細のいずれか、またはすべてを組み込むことができます。
  - データ型
  - 名前のクラス
  - 名前の相対アドレス
  - アラインメント
  - 次元
  - 配列の場合は、それが割り振り可能かどうか
  - それがポインター、ターゲット、整数ポインターのいずれであるか
  - それがパラメーターであるかどうか
  - それが揮発性であるかどうか
  - 仮引数について、その意図、それが値であるかどうか、それが任意指定かどうか
  - private、public、protected、モジュール
- エンティティーを定義、参照、あるいは変更した場所を示すための座標。エンティティーを宣言すると、座標に `$` マークが付きます。エンティティーを初期化すると、座標に `*` マークが付きます。同じ場所でエンティティーの宣言および初期化の両方を行うと、座標に `&` マークが付きます。エンティティーが設定されると、座標に `@` マークが付きます。エンティティーが参照されると、座標には何もマークが付きません。

-qxref または **-qattr** によって**完全な**サブオプションを指定すると、C および C++ はコンパイル単位内のすべてのエンティティーを報告します。このサブオプションを指定しないと、実際に使用しているエンティティーだけが表示されます。

オブジェクト・セクション

XL C/C++ は、**-qlist** コンパイラー・オプションが有効になっている場合にのみ、このセクションを作成します。このセクションには、ソース行番号、命令オフセット (16 進表記)、命令のアセンブラー・ニーモニック、命令の 16 進値を示す疑似アセ

ンブラー・オブジェクト・コード・リストが入っています。右側には、命令のサイクル・タイムとコンパイラーの中間言語も示されます。最後に、合計サイクル・タイム (直線的実行時間) と、作成されたマシン・インストラクションの合計数が表示されます。コンパイル単位ごとに別個のセクションがあります。

注: 真のアセンブラー・リストを入手するには、ユーザーのアプリケーションをコンパイルするときに、**-S** コンパイラー・オプションを指定します。アセンブラー・リストには、*filename*.s という名前が付けられます。

ファイル・テーブル・セクション

このセクションには、使用されている各メイン・ソース・ファイルとインクルード・ファイルのファイル番号とファイル名を示すテーブルが含まれています。また、インクルード・ファイルが参照されるメイン・ソース・ファイルの行番号もリストします。このセクションは必ず存在します。

コンパイル単位エピローグ・セクション

これは、各コンパイル単位のリストの最後のセクションです。これには診断の要約が含まれていて、その単位が正常にコンパイルされたかどうかを示します。ファイルにコンパイル単位が 1 つしか含まれていない場合は、このセクションはリストに存在しません。

コンパイル・エピローグ・セクション

コンパイル・エピローグ・セクションは、リストの終わりに 1 回発生するだけです。コンパイルの完了時に、XL C/C++ はコンパイルの概要を提示します。概要とは、読み取られたソース・レコードの数、コンパイル開始時刻、コンパイルの終了時刻、合計コンパイル時間、合計 CPU 時間、および仮想 CPU 時間です。このセクションは、リストに必ず存在します。

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。次に、 **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、コンパイル済みアプリケーションをステップスルーしその動作を検査することができます。

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す。

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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
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。

IBM Corporation
Lab Director
IBM Canada Limited
8200 Warden Avenue
Markham, Ontario, Canada
L6G 1C7

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。

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。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

注: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。

AIX	IBM	OS/390
OS/400	POWER3	POWER4
POWER5	POWER5+	PowerPC
z/OS	z/VM	

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

OpenMP は、OpenMP Architecture Review Board の商標です。

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。

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

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

索引

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

[ア行]

アーカイブ・ファイル 26
アセンブラー
  ソース (.S) ファイル 26
  ソース (.s) ファイル 26, 27
エピローグ・セクション、コンパイラー・リストの 38
エラー・メッセージ
  形式の説明 35
  コンパイラー・リストの 35
オブジェクト・ファイル 26, 27
オプション・セクション、コンパイラー・リストの 35
オンライン資料 1
オンライン・コンパイラー・ヘルプ 1

[カ行]

環境変数
  コンパイル時 21
  ランタイム 21
共用オブジェクト・ファイル 26
共用メモリーの並列処理 8
警告エラー 36
言語サポート 3
コードの最適化 6
構成ファイル 22, 26
構成ファイルのカスタマイズ (デフォルトのコンパイラー・オプションも含む) 22
コマンド行オプション
  参照： コンパイラー・オプション
コンパイラーの稼働 24
コンパイラーの実行 24
コンパイラーの呼び出し 24
コンパイラー・オプション
  セクション、コンパイラー・リストの 35
  参照： 『特殊文字』の項にリストされている個々のオプション
コンパイラー・リスト 34
コンパイル
  プログラムをコンパイルする方法の説明 24
コンパイル *(続き)*
  SMP プログラム 25
コンパイル単位エピローグ・セクション、コンパイラー・リストの 38

[サ行]

最適化 6
実行、プログラムの 31
実行可能ファイル 27
重大エラー 36
出力ファイル 27
資料、オンライン形式 1
シンボリック・デバッガー・サポート 9
静的リンク 30
ソース・コード規格合致検査 4
ソース・セクション、コンパイラー・リストの 35
ソース・ファイル 26
ソース・ファイルの編集 23
ソース・レベル・デバッグ・サポート 9
相互参照セクション、コンパイラー・リストの 37
属性セクション、コンパイラー・リストの 37

[タ行]

ツール 5
  インストール・ユーティリティー 5
  構成ファイル・ユーティリティー 5
  新規のインストール構成ユーティリティー 5
  cleanpdf 5
  gxlc および gxlc++ 6
  mergepdf 5
  new_install 5
  resetpdf 6
  showpdf 6
  vac_configure 5
  xlc_install 5
通知メッセージ 36
テキスト・エディター 23
デバッガー・サポート 9, 38
デバッグ 33, 38
デフォルト
  コンパイラー・デフォルトのカスタマイズ 22
動的リンク 30

[ナ行]

入力ファイル 26

[ハ行]

パラメーター
  参照： 引数
ファイル
  出力 27
  ソースの編集 23
  入力 26
ファイル・テーブル・セクション、コンパイラー・リストの 38
プロファイル、データ・ファイルの 27
並列処理 8
ヘッダー・セクション、コンパイラー・リストの 34
ヘルプ・システム 1
変換レポート・セクション、コンパイラー・リストの 37

[マ行]

マルチプロセッサー・システム 8
問題判別 33

[ヤ行]

呼び出し、プログラムの 31

[ラ行]

ライブラリー 26
ランタイム
  ライブラリー 26
ランタイム環境 31
ランタイム・オプション 31
リカバリー不能エラー 36
リスト・ファイル 27
リンク 29
  静的 30
  動的 30
例のプログラム
  参照： サンプル・プログラム

［数字］

15xx は XL C/C++ メッセージの ID 36
64 ビット環境 7

A

ANSI
　C/C++ 規格への合致の検査 4
a.out ファイル 27

E

E のエラーの重大度 36

H

HTML 資料 1

I

I のエラーの重大度 36
ISO
　C/C++ 規格への合致の検査 4
I/O
　参照： 入出力

M

Make ファイル
　デフォルト・オプションの代替として
　　の構成ファイル 22
mod ファイル 26, 27

O

OpenMP 8

P

PDF 資料 1

S

S のエラーの重大度 36
SMP
　プログラム、コンパイル 25
SMP プログラム 8

U

U のエラーの重大度 36

W

W のエラーの重大度 36

X

XL C/C++ のレベル、判別 22

［特殊文字］

-qattr コンパイラー・オプション 37
-qlist コンパイラー・オプション 37
-qlistopt コンパイラー・オプション 35
-qreport コンパイラー・オプション 37
-qsource コンパイラー・オプション 35
-qxref コンパイラー・オプション 37
.a ファイル 26
.c および .C ファイル 26
.cfg ファイル 26
.lst ファイル 27
.mod ファイル 26, 27
.o ファイル 26, 27
.S ファイル 26
.s ファイル 26, 27
.so ファイル 26
/etc/opt/ibmcmp/vacpp/8.0/vac.cfg 構成ファ
　イル 22

プログラム番号: 5724-M16

SD88-6730-00

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