インテル® Fortran コンパイラー 11.1 Mac OS* X 版 プロフェッショナル・エディション インストール・ガイドおよび リリースノート

資料番号: 321416-002JA
2009 年 7 月

目次

1 概要 3
1.1 変更履歴 3
1.2 製品の内容 3
1.3 動作環境 3
1.4 ドキュメント 3
1.5 テクニカルサポート 3
2 インストール 4
2.1 インストール先フォルダー 4
2.2 インストール後の製品の配置変更 5
2.3 削除/アンインストール 5
3 インテル® Fortran コンパイラー 5
3.1 互換性 5
3.1.1 型バインド・プロシージャーの誤った派生型レイアウト 6
3.2 新機能と変更された機能 6
3.2.1 Fortran 2003 の機能 6
3.2.2 その他の変更 6
3.3 新規および変更されたコンパイラー・オプション 6
3.3.1 -O0 は -mp を含みません 7
3.4 その他の変更 7
3.4.1 最適化レポートがデフォルトで無効 7
3.4.2 I/O を制御する新しい環境変数 7
3.4.3 環境設定スクリプトの変更 7
3.5 既知の問題 8
3.5.1 空の派生型の制限付きサポート 8
3.6 Fortran 2003 機能の概要 9
4 インテル® デバッガー (IDB) 11

4.1 既知の問題 11
4.1.1 Dwarf と Stabs デバッグ・フォーマット 11
4.1.2 コンパイル要件 11
4.1.3 非ローカルのバイナリーファイルとソースファイルのアクセス 11
4.1.4 ローカル変数は表示されません 12
4.1.5 Fortran REAL*16 変数の出力 12
4.1.6 fork アプリケーションのデバッグ 12
4.1.7 exec アプリケーションのデバッグ 12
4.1.8 Fortran 代替エントリーポイント 12
4.1.9 スナップショット 12
4.1.10 最適化コードのデバッグ 12
4.1.11 ウォッチポイント 12
4.1.12 Fortran モジュールと common 13
4.1.13 グラフィック・ユーザー・インターフェイス (GUI) 13
4.1.14 MPP デバッグの制限 13
4.1.15 関数ブレークポイント 13
4.1.16 コアファイルのデバッグ 13
4.1.17 ユニバーサル・バイナリーのサポート 13
4.1.18 $threadlevel デバッガー変数 13
4.1.19 オープンファイル記述子の制限 14
4.1.20 $cdir ディレクトリー、$cwd ディレクトリー 14
4.1.21 `info stack` の使用 14
4.1.22 `$stepg0` のデフォルト値が変更 14
5 インテル® マス・カーネル・ライブラリー 14
5.1 本バージョンでの変更 15
5.1.1 新機能 15
5.1.2 ユーザービリティー/インターフェイスの向上 15
5.1.3 パフォーマンスの向上 15
5.2 既知の問題 16
5.3 注意事項 17
5.4 権利の帰属 17
6 著作権と商標について 18

1 概要

このドキュメントでは、製品のインストール方法、新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題について説明します。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。報告されている問題の修正リストは、インテル® ソフトウェア開発製品レジストレーション・センターで提供されている各アップデート製品に含まれる README.TXT ファイルを参照してください。

Update 1

- 型バインド・プロシージャーを含む派生型を宣言または使用しているソースの再コンパイルの必要性
- `-O0` の動作の変更に関する注意事項の追加
- `FORT_BLOCKSIZE` 環境変数と `FORT_BUFFERCOUNT` 環境変数の記載
- 報告されている問題の修正

1.2 製品の内容

インテル® Fortran コンパイラー 11.1 Mac OS X 版プロフェッショナル・エディション* には、次のコンポーネントが含まれています。

- インテル® プロセッサー・ベースの Mac OS X オペレーティング・システムで動作するアプリケーションのビルド用インテル® Fortran コンパイラー
- インテル® デバッガー
- インテル® マス・カーネル・ライブラリー (インテル® MKL)
- Xcode* 開発環境への統合 (制限付きの機能)
- 各種ドキュメント

1.3 動作環境

- インテル® プロセッサー・ベースの Apple* Mac* システム
- RAM 1GB (最小)、RAM 2GB (推奨)
- 2GB のディスク空き容量
- Mac OS X 10.5.6 と Xcode 3.1.2 または Mac OS X 10.5.7 と Xcode 3.1.3
- gcc* 4

注: 高度な最適化オプションを使用する場合や大規模なプログラムの場合、メモリーやディスク容量など、追加でリソースが必要になることがあります。

1.4 ドキュメント

製品ドキュメントは、「インストール先フォルダー」で示されているように、`Documentation` フォルダーに保存されています。

1.5 テクニカルサポート

インストール時にコンパイラーの登録を行わなかった場合は、インテル® ソフトウェア開発製品レジストレーション・センターで登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

テクニカルサポート、製品のアップデート、ユーザーフォーラム、FAQ、ヒント、およびその他のサポート情報は、http://www.intel.com/software/products/support/ (英語) を参照してください。

注: 代理店がテクニカルサポートを提供している場合は、インテルではなく代理店にお問い合わせください。

2 インストール

初めて製品をインストールする場合は、インストール中にシリアル番号の入力が求められますので、あらかじめご用意ください。製品のインストールと使用には、有効なライセンスが必要です。

Xcode を使用する場合、サポートされているバージョンの Xcode がインストールされていることを確認してください。将来、新しいバージョンの Xcode をインストールする場合は、そのインストール後にインテル® Fortran コンパイラーを再インストールする必要があります。

製品のインストール、変更、アンインストールを行うには、管理者権限または "sudo" 権限が必要です。

DVD 版の場合は、DVD を挿入し、 DVD のディスク・イメージ・ファイル (`m_cprof_p_11.1.xxx.dmg`) まで移動してダブルクリックします。ダウンロード版の場合は、ダウンロード・ファイル (`m_cprof_p_11.1.xxx.dmg`) をダブルクリックします。

手順に従ってインストールを完了します。

2.1 インストール先フォルダー

11.1 製品は、前のバージョンとは異なる構成でフォルダーにインストールされます。新しい構成を以下に示します。一部含まれていないフォルダーもあります。

- `<root>/intel/Compiler/11.1/`*`xxx`*`/`
  - `bin`
    - `ia32`
    - `intel64`
  - `include`
    - `ia32`
    - `intel64`
  - `lib`
  - `Frameworks`
    - `mkl`
  - `Documentation`
  - `man`
  - `Samples`

`<root>` はデフォルトでは `/opt`、 *`xxx`* は 3 桁のアップデート番号です。`bin`、`include`、`lib` 配下のフォルダーは次のとおりです。

- `ia32`: 32 ビットのインテル® プロセッサー・ベースの Mac OS X システム上で動作するアプリケーションのビルド用コンパイラー
- `intel64`: 64 ビットのインテル® プロセッサー・ベースの Mac OS X システム上で動作するアプリケーションのビルド用コンパイラー (インテル® 64 アーキテクチャーとも呼ばれます)

インテル® C++ コンパイラーとインテル® Fortran コンパイラーの両方がインストールされている場合、所定のバージョンのフォルダーが共有されます。

2.2 インストール後の製品の配置変更

付属のスクリプトを使用して、インストールされた製品のコマンドライン・インターフェイスをディスクの別の場所に移動することができます。

1. 端末を開きます。
2. コンパイラーのインストール・フォルダーに移動 (cd) します (例: `/opt/intel/Compiler/11.1/xxx`)。
3. コマンドを入力します。
   `./move_cprof.sh <new-install-location>`
   `<new-install-location>` は新規のディレクトリー・パスです。

このスクリプトはすべてのファイルを移動し、必要に応じてシンボリック・リンク、環境変数、起動スクリプトを更新します。インテル® C++ コンパイラーとインテル® Fortran コンパイラーの両方が古いパスにインストールされている場合は、両コンパイラーとも新しい場所に移動されます。

Xcode 統合は、Xcode ディレクトリー・ツリーを別の場所にドラッグアンドドロップするだけで移動できます。移動した Xcode ディレクトリー・ツリーを使用してコマンドプロンプトから `idb` を使用する場合は、http://software.intel.com/en-us/articles/running-idb-from-command-line-after-relocating-xcode-environment/ (英語) を参照して、必要なその他の手順を確認してください。`idb` は Xcode IDE 内では利用できないことに注意してください。

2.3 削除/アンインストール

パフォーマンス・ライブラリー・コンポーネントを残してコンパイラーのみを削除することはできません。

1. 端末を開いて、`<install-dir>` 以外のフォルダーに移動 (`cd`) します。
2. 次のコマンドを入力します: `<install-dir>/uninstall_cprof.sh`
3. 画面の指示に従ってオプションを選択します。

`root` でログインしていない場合は `root` パスワードの入力が求められます。同じバージョンのインテル® C++ コンパイラーをインストールしている場合は、C++ コンパイラーも削除されます。

3 インテル® Fortran コンパイラー

このセクションでは、インテル® Fortran コンパイラーの変更点、新機能、および最新情報をまとめています。

3.1 互換性

一般に、インテル® Fortran コンパイラー Mac OS X 版の以前のバージョンでコンパイルされたオブジェクト・コードおよびモジュールは、バージョン 11.1 でもそのまま使用できます。ただし、次の例外があります。

- マルチファイルのプロシージャー間の最適化 (`-ipo`) オプションを使用してビルドされたオブジェクトは、再コンパイルする必要があります。
- バージョン 9.1 のコンパイラーを使用して 64 ビット・システム用にビルドされた、REAL(16) または REAL*16 データ型を使用するオブジェクトは再コンパイルする必要があります。
- バージョン 11 よりも前のコンパイラーを使用してコンパイルされた、ATTRIBUTES ALIGN 宣言子を指定したモジュールは再コンパイルする必要があります。この問題が発生した場合、問題を通知するメッセージが表示されます。

3.1.1 型バインド・プロシージャーの誤った派生型レイアウト

初期のバージョンの 11.1 コンパイラーでは未使用の空間が型バインド・プロシージャーを含む派正型に誤って追加されます。この問題は 11.1 Update 1 で修正されています。初期の 11.1 コンパイラーでコンパイルされたそのような型のオブジェクトを宣言または使用するすべてのソースは、バージョン 11.1 Update 1 以降を使用して再コンパイルする必要があります。

3.2 新機能と変更された機能

いくつかの言語機能についての説明がコンパイラー・ドキュメントにまだ含まれていない可能性があります。必要に応じて、Fortran 2003 規格 (http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf) を参照してください。

3.2.1 Fortran 2003 の機能

- オブジェクト指向の機能
  - CLASS 宣言
  - SELECT TYPE 構造
  - EXTENDS_TYPE_OF と SAME_TYPE_AS 組み込み関数
  - 多相型エンティティー
  - 継承と関連付け
  - 遅延バインディングと抽象型
  - 型問い合わせ組み込み関数
- 型バインド・プロシージャー
  - TYPE CONTAINS 宣言
  - ABSTRACT 属性
  - DEFERRED 属性
  - NON_OVERRIDABLE 属性
  - **注:** GENERIC 属性と型バインド操作はこのリリースではサポートされていません。
- 無指定文字長エンティティー
- PRIVATE コンポーネントの PUBLIC 型と PUBLIC コンポーネントの PRIVATE 型
- NAMELIST I/O が内部ファイルで許可
- NAMELIST グループのエンティティーの制限の緩和
- 書式付き入出力で IEEE 無限大と NaN の表現方法が変更
- SYSTEM_CLOCK 組み込み関数の COUNT_RATE 引数が任意の種類の REAL で指定可能
- STOP 文の実行で IEEE 浮動小数点例外が発生すると警告を表示
- `-assume noold_maxminloc` が指定された場合、ゼロサイズの配列の MAXLOC または MINLOC でゼロを返します。Fortran 95 では値はプロセッサー依存で、インテル® Fortran は 1 を返していました。`-assume noold_maxminloc` が指定されるとパフォーマンスが低下します。

3.2.2 その他の変更

- 文字列長チェックが有効 (`-check bounds`) で文字オブジェクトが引数として渡されると、渡された長さの最小値と呼び出されたプロシージャーで宣言された長さが上限として使用されます。
- リストで対応する変数が LOGICAL ではない場合、リスト指定またはネームリスト指定の入力で、LOGICAL 定数形式の入力値項目 (例: T または .F) は許可されなくなりました。新しい `-assume old_logical_ldio` オプションを使用して以前の動作に戻すことができます。
- 浮動小数点例外動作のコンパイルごとの制御 (`-fpe-all`)

3.3 新規および変更されたコンパイラー・オプション

詳細は、コンパイラーのドキュメントを参照してください。

- -assume [no]old_logical_ldio
- -assume [no]old_maxminloc
- -fpe-all
- -ieee_fpe_flags
- -mkl[=lib]

廃止予定のコンパイラー・オプションのリストは、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.3.1 -O0 は -mp を含みません

バージョン 11.1 では、最適化を無効にする `-O0` オプションが 浮動小数点精度を最大化する `-mp` を含意しなくなりました。`-mp` スイッチは廃止予定です。そのため、浮動小数点精度の影響を受けやすいアプリケーションには、明示的に `-fp-model` オプションを指定することを推奨します。

3.4 その他の変更

3.4.1 最適化レポートがデフォルトで無効

バージョン 11.1 以降、コンパイラーは、ベクトル化、自動並列化、OpenMP スレッド化ループに関する最適化レポートメッセージをデフォルトで表示しないようになりました。これらのメッセージを表示するには、`-diag-enable vec`、`-diag-enable par`、`-diag-enable openmp` を指定するか、`-vec-report`、`-par-report`、`-openmp-report` を使用する必要があります。

また、バージョン 11.1 以降、最適化レポートメッセージは `stdout` ではなく、`stderr` に送られます。

3.4.2 I/O を制御する新しい環境変数

バージョン 11.1 では、アプリケーションの実行時に I/O 動作に影響する 2 つの環境変数が追加されています。

`FORT_BLOCKSIZE` は、`OPEN` 文で `BLOCKSIZE=` が省略されたときに使用されるデフォルトの `BLOCKSIZE` 値を指定できます。有効なサイズは 0 から 2147467264 までの値です。サイズは最も近い 512 バイト境界に丸められます。デフォルトの `BLOCKSIZE` 値は 128KB です。

`FORT_BUFFERCOUNT` は、`OPEN` 文で `BUFFERCOUNT=` が省略されたときに使用されるデフォルトの `BUFFERCOUNT` 値を指定できます。有効な値は 0 から 127 までの値です。0 が指定された場合、デフォルト値の 1 が使用されます。

3.4.3 環境設定スクリプトの変更

コマンドライン・ビルド環境の設定に使用されていた ifortvars.sh (ifortvars.csh) スクリプトが変更されました。以前のバージョンでは、fc または fce のいずれかのルート・ディレクトリーを選択することによってターゲット・プラットフォームが選択されました。バージョン 11.x では、スクリプトは 1 つのみで、引数を指定してターゲット・プラットフォームを選択します。

コマンドの形式は以下のとおりです。

```
source /opt/intel/Compiler/11.1/xxx/bin/ifortvars.sh argument
```

xxx はリビジョン番号です。*argument* は `ia32` または `intel64` のいずれかです (「インストール先フォルダー」を参照)。コンパイラーを異なるパスにインストールしている場合

は、適切なフォルダーを指定しください。コンパイラー環境を構築すると、インテル® デバッガー (idb) 環境も構築されます。

3.5 既知の問題

3.5.1 空の派生型の制限付きサポート

Fortran 2003 では、派生型をデータ・コンポーネントなしで宣言する機能が追加されています。インテル® コンパイラーの現在のリリースでは、このサポートには制限があります。将来のリリースでこの制限は解除される予定です。制限事項は次のとおりです。

- 派生型のオブジェクトが宣言される際、型には少なくとも 1 つのデータ・コンポーネントがなければなりません。空の型の拡張がサポートされています。次に例を示します。

```
type t
end type

type, extends (t)  :: t1
end type

type, extends (t1) :: t2
  integer i
end type

type, extends (t2) :: t3
end type

type (t) :: rec1 ! Not supported, type t is empty
type (t1) :: rec2 ! Not supported, type t1 is empty
type (t2) :: rec3 ! Supported, type t2 is not empty
type (t3) :: rec4 ! Supported, type t3 is not empty
```

  例外として、クラスオブジェクトの空の型での宣言はサポートされています。次に例を示します。

```
class(t1) :: rec5
```

  サポートされていない空の型が見つかると、次のメッセージが表示されます。

  `Declaring an object with no data component fields is not yet supported` (データ・コンポーネント・フィールドのないオブジェクトの宣言はサポートされていません)

- 空の型のコンポーネントの参照はサポートされていません。例えば、上記の宣言を仮定します。

```
call sub(rec4%t3, rec4%t1, rec3%t)
print *, rec3%t1, rec4%t
call sub2(rec3%t2, rec4%t2)
```

  `rec4%t3`、`rec4%t1`、`rec4%t`、`rec3%t1`、`rec3%t` への参照はサポートされていません。`rec3%t2` と `rec4%t2` の参照はサポートされます。サポートされていない参照が見つかると、次のメッセージが表示されます。

  `Accessing an empty type is not yet supported` (空の型へのアクセスはサポートされていません)

- 空の型の型コンストラクターはサポートされていません。上記の宣言を例にとると、型コンストラクター `t()` はサポートされていません。サポートされていないコンストラクターが見つかると、次のメッセージが表示されます。

  `A type constructor for an empty type is not yet supported` (空の型の型コンストラクターはサポートされていません)

3.6 Fortran 2003 機能の概要

インテル® Fortran コンパイラーは、最新の Fortran 規格である、Fortran 2003 の多くの機能をサポートしています。現在サポートしていない Fortran 2003 機能についても、今後サポートしていく予定です。現在のコンパイラーでは、以下の Fortran 2003 機能がサポートされています。

- Fortran 文字セットが次の 8 ビット ASCII 文字を含むように拡張: ~ \ [ ] ` ^ { } | # @
- 最大長 63 文字までの名前
- 最大 256 行の文
- 角括弧 [ ] を (/ /) の代わりに配列の区切り文字として使用可能
- コンポーネント名とデフォルト初期化を含む構造コンストラクター
- 型と文字列長仕様を含む配列コンストラクター
- 名前付き PARAMETER 定数は複素定数の一部
- 列挙子
- 割り当て可能な派生型のコンポーネント
- 割り当て可能なスカラー変数
- 無指定文字長エンティティー
- PRIVATE コンポーネントの PUBLIC 型と PUBLIC コンポーネントの PRIVATE 型
- ALLOCATE と DEALLOCATE の ERRMSG キーワード
- ALLOCATE の SOURCE= キーワード
- 型拡張子
- CLASS 宣言
- 多相型エンティティー
- 継承と関連付け
- 遅延バインディングと抽象型
- 型バインド・プロシージャー
- TYPE CONTAINS 宣言
- ABSTRACT 属性
- DEFERRED 属性
- NON_OVERRIDABLE 属性
- ASYNCHRONOUS 属性および文
- BIND(C) 属性および文
- PROTECTED 属性および文
- VALUE 属性および文
- VOLATILE 属性および文
- ポインター・オブジェクトの INTENT 属性
- 代入文の左辺と右辺の形状または長さが異なる場合に、左辺の割り当て可能な変数を再割り当て ("assume realloc_lhs" オプションが必要)
- ASSOCIATE 構造
- SELECT TYPE 構造
- すべての I/O 文で、次の数値は任意の種類で指定可能: UNIT=, IOSTAT=
- NAMELIST I/O が内部ファイルで許可
- NAMELIST グループのエンティティーの制限の緩和
- 書式付き入出力で IEEE 無限大と NaN の表現方法が変更
- FLUSH 文
- WAIT 文

- OPEN の ACCESS='STREAM' キーワード
- OPEN およびデータ転送文の ASYNCHRONOUS キーワード
- INQUIRE およびデータ転送文の ID キーワード
- データ転送文の POS キーワード
- INQUIRE の PENDING キーワード
- 次の OPEN 数値は任意の種類で指定可能: RECL=
- 次の READ および WRITE 数値は任意の種類で指定可能: REC=, SIZE=
- 次の INQUIRE 数値は任意の種類で指定可能: NEXTREC=, NUMBER=, RECL=, SIZE=
- 開始する新しい I/O が自身以外の内部ファイルを修正しない内部 I/O の場合、再帰 I/O を利用可能
- IEEE 無限大および非数は Fortran 2003 で指定されるフォーマット出力で表示
- BLANK、DECIMAL、DELIM、ENCODING、IOMSG、PAD、ROUND、SIGN、SIZE I/O キーワード
- DC、DP、RD、RC、RN、RP、RU、RZ 書式編集記述子
- I/O フォーマットで、繰り返し指定子が続く場合、P 編集記述子の後のカンマはオプション
- USE 内のユーザー定義演算子名の変更
- USE の INTRINSIC および NON_INTRINSIC キーワード
- IMPORT 文
- 割り当て可能なダミー引数
- 割り当て可能な関数結果
- PROCEDURE 宣言
- プロシージャー・ポインター
- ABSTRACT INTERFACE
- PASS 属性と NOPASS 属性
- SYSTEM_CLOCK 組み込み関数の COUNT_RATE 引数が任意の種類の REAL で指定可能
- STOP 文の実行で IEEE 浮動小数点例外が発生すると警告を表示
- `-assume noold_maxminloc` が指定された場合、ゼロサイズの配列の MAXLOC または MINLOC でゼロを返します。
- 型問い合わせ組み込み関数
- COMMAND_ARGUMENT_COUNT 組み込み関数
- EXTENDS_TYPE_OF と SAME_TYPE_AS 組み込み関数
- GET_COMMAND 組み込み関数
- GET_COMMAND_ARGUMENT 組み込み関数
- GET_ENVIRONMENT_VARIABLE 組み込み関数
- IS_IOSTAT_END 組み込み関数
- IS_IOSTAT_EOR 組み込み関数
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC 組み込み関数 (CHARACTER 引数)
- MOVE_ALLOC 組み込み関数
- NEW_LINE 組み込み関数
- SELECTED_CHAR_KIND 組み込み関数
- 次の組み込み関数はオプションの KIND= 引数を使用: ACHAR, COUNT, IACHAR, ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE, UBOUND, VERIFY
- ISO_C_BINDING 組み込みモジュール
- IEEE_EXCEPTIONS、IEEE_ARITHMETIC、IEEE_FEATURES 組み込みモジュール
- ISO_FORTRAN_ENV 組み込みモジュール

サポートされていない Fortran 2003 機能には次の項目が含まれます。

- 型バインド・プロシージャーの型バインド操作と GENERIC バインド
- ユーザー定義の派生型 I/O
- パラメーター化された派生型

4 インテル® デバッガー (IDB)

4.1 既知の問題

4.1.1 Dwarf と Stabs デバッグ・フォーマット

デバッガーでは、デバッグ情報が Dwarf フォーマットの実行ファイルのデバッグのみをサポートしており、Stabs デバッグ・フォーマットはサポートしていません。gcc と g++ で Dwarf 出力を生成するには、コンパイルコマンドで `-gdwarf-2` フラグを使用します。インテル® コンパイラー (icc と ifort) では、`-g` フラグで Dwarf デバッグ・フォーマットを作成します。

4.1.2 コンパイル要件

Xcode 2.3 より、Dwarf デバッグ情報はオブジェクト (.o) ファイルに保存されています。これらのオブジェクト・ファイルは、デバッグ対象のアプリケーションに関連した情報を得るためにデバッガーによりアクセスされます。そのため、シンボリック・デバッグが利用可能でなければなりません。

次のように、1 つのコマンドでプログラムがコンパイルされ、リンクされた場合、

```
ifort -g -o hello.exe hello.f90
```

コンパイラーによりオブジェクト・ファイルは生成されますが、コマンドが完了する前に削除されます。このコマンドで作成されたバイナリーファイルにはデバッグ情報は含まれません。アプリケーションをデバッグ可能にするには、次の 2 つの方法があります。

アプリケーションを 2 つの手順でビルドして .o ファイルを明示的に作成します。

```
ifort -c -g -o hello.o hello.f90

ifort -g -o hello.exe hello.o
```

または、-save-temps コンパイラー・スイッチを使用して作成された .o ファイルが削除されないようにします。

```
ifort -g -save-temps -o hello.exe hello.f90
```

デバッガーは "dsymutil" ユーティリティーの出力を使用しません。

4.1.3 非ローカルのバイナリーファイルとソースファイルのアクセス

デバッガーは、ネットワーク・マウント・ファイル・システム (NFS など) からバイナリーファイルにアクセスできません。次のようなエラーメッセージが表示されます。

Internal error: cannot create absolute path for: /home/me/hello (内部エラー: /home/me/hello の絶対パスを作成できません。)

You cannot debug "/home/me/hello" because its type is "unknown". ("/home/me/hello" はデバッグできません。型が "不明" です。)

また、デバッガーは、ネットワーク・マウント・ファイル・システム (NFS など) からソースファイルにアクセスできません。次のようなエラーメッセージが表示されます。

Source file not found or not readable, tried... (ソースファイルが見つからないか読み取りできません...)

./hello.f90

/auto/mount/site/foo/usr1/user_me/f_code/hello.f90

(Cannot find source file hello.f90 (ソースファイル hello.f90 が見つかりません))

The file-path specified will be correct. (指定されたファイルパスは修正されます。)

ファイルは、ローカル・ファイル・システム (例: ネットワークでマウントされていないシステム) にコピーして使用してください。

4.1.4 ローカル変数は表示されません

Mac OS X 10.5.4 (および以降のバージョン) のリンカーでは、ローカル変数の定義を実行ファイルのデバッグ情報に常に出力するわけではありません。ローカル変数の定義がいつデバッグ情報に出力されるかが不明です。その結果、変数が表示されなかったり、表示されても評価が適切でない場合があります。

これまでの事例では、コンパイラーによって `.bss` セグメントに割り当てられた Fortran プログラムのローカル配列が関与しています。回避方法として、変数をローカルではなくグローバルにするようソースを変更します。Fortran では変数をモジュールまたは共通ブロックに置くことでこれを簡単に行えます。インテル社とアップル社では協力してこの問題の解決に取り組んでいます。

4.1.5 Fortran REAL*16 変数の出力

デバッガーでは、Fortran REAL*16 変数の正しい値が出力されません。

4.1.6 fork アプリケーションのデバッグ

fork を呼び出すアプリケーションの子プロセスのデバッグはまだサポートされていません。

4.1.7 exec アプリケーションのデバッグ

$catchexecs 制御変数はサポートされていません。

4.1.8 Fortran 代替エントリーポイント

代替エントリーポイントの仮パラメーターは、メイン・エントリー・ポイントでも仮パラメーターでなければデバッガー内では見えません。

4.1.9 スナップショット

マニュアルで説明されているスナップショットはまだサポートされていません。

4.1.10 最適化コードのデバッグ

最適化コードのデバッグはまだ完全にはサポートされていません。最適化を有効にしてコードをコンパイルすると、一部の関数名、パラメーター、変数、パラメーターと変数の内容をデバッガーが参照できないことがあります。

4.1.11 ウォッチポイント

書き込みアクセスを検知するよう作成されたウォッチポイントは、元の値と同一の値が書き込まれたときにはトリガーしません。これは、Mac OS X オペレーティング・システムの制限によるものです。

ウォッチポイントの実装には、SIGSEGV シグナルではなく SIGBUS シグナルがデバッガーで使用されているため、SIGBUS シグナルをキャッチするシグナル・ディテクターを作成することができません。

4.1.12 Fortran モジュールと common

グローバルに定義された Fortran モジュールは参照時にパーセント記号 2 つ (%%) でスコープを変更します。たとえば、グローバルに定義されたモジュール foo に含まれるサブルーチン bar にブレークポイントを設定するには、次のように指定します。

```
(idb) stop in foo%%bar
```

構文についてはマニュアルの次のセクションを参照してください。

Looking Around the Code, the Data and Other Process Information >

Looking at the Data >

The print Command

ソースコードの名前を使用して Fortran モジュールや common にアクセス (print など) しようとすると、デバッガーで名前を認識できないことがあります。回避策として、名前の先頭に '_' を追加します。たとえば、ソースコードで "com" という common がある場合は次のように指定します。

```
(idb) print _com
```

4.1.13 グラフィック・ユーザー・インターフェイス (GUI)

本バージョンのデバッガーでは GUI はサポートされていません。

4.1.14 MPP デバッグの制限

マニュアルで説明されている MPP デバッグはサポートされていません。

4.1.15 関数ブレークポイント

関数に設定されたブレークポイント ("stop in" コマンドを使用して設定) では、最初の文でユーザープログラムの実行が停止されないことがあります。これは、生成された Dwarf デバッグ情報で関数プロローグに関する情報が不十分なために発生します。回避策として、"stop at" コマンドで該当文にブレークポイントを設定します。

コンパイラーは "__dyld_func_lookup" への呼び出しを関数のプロローグの一部として作成します。この関数にブレークポイントを設定すると、デバッガーはその位置で停止しますが、ローカル変数値が有効ではありません。回避策として、ブレークポイントを関数内の最初の文に設定します。

4.1.16 コアファイルのデバッグ

コアファイルのデバッグは、サポートされていません。

4.1.17 ユニバーサル・バイナリーのサポート

ユニバーサル・バイナリーのデバッグはサポートされています。デバッガーは IA-32 上の IA-32 Dwarf セクションのバイナリーと、インテル® 64 上の IA-32 セクションまたはインテル® 64 セクションのデバッグをサポートしています。

4.1.18 $threadlevel デバッガー変数

マニュアルでは、"$threadlevel" デバッガー変数について「On Mac OS* X, the debugger supports POSIX threads, also known as pthreads. (Mac OS* X では、デバッガーは POSIX スレッド (pthreads とも呼ばれる) をサポートしています。」という記述があります。この文章で

は別の種類のスレッドもサポートされているようにもとれますが、 そうではなく、POSIX スレッドのみが Mac OS X 上でサポートされています。

4.1.19 オープンファイル記述子の制限

デバッガーはデバッグ対象の .o ファイルを開いてデバッグ情報を読み取るため、ファイルの制限を緩和する必要があります。

Mac OS では、オープンできるファイル記述子の数を 256 に制限していますが、 次のように上限を上げることができます。

```
ulimit -n 2000
```

デバッガーを起動する前に、このコマンドを使用してオープンファイル記述子の数の制限を上げてください。

これは、デバッガーが多くのファイルに対してオープンファイル記述子の制限数を適切に共有できるようになるまでの回避策です。

4.1.20 $cdir ディレクトリー、$cwd ディレクトリー

`$cdir` はコンパイル・ディレクトリーです (記録されている場合)。`$cdir` は、ディレクトリーが設定されている場合にサポートされます。シンボルとしてサポートされるわけではありません。

`$cwd` は現在の作業ディレクトリーです。セマンティクスもシンボルもサポートされていません。

`$cwd` と '.' の違いは、`$cwd` はデバッグセッション中に変更された現在の作業ディレクトリーを追跡する点です。'.' は、ソースパスへのエントリーが追加されると直ちに現在のディレクトリーに展開されます。

4.1.21 `info stack` の使用

デバッガーコマンド "info stack" は、以下のオプションの構文では現在、負のフレームカウントをサポートしていません。

```
  info stack [num]
```

フレームカウント num が正の場合、最内 num フレームを出力します。カウントが負またはゼロの場合、(最外 num フレームを出力するのではなく) フレームを出力しません。

4.1.22 `$stepg0` のデフォルト値が変更

デバッガー変数 `$stepg0` のデフォルト値が 0 に変更されました。この設定では、"step" コマンドを使用する場合、デバッガーはデバッグ情報なしでコードにステップインします。以前のデバッガーバージョンと互換性を保つようにするには、次のようにデバッガー変数を 1 に設定します。

```
(idb) set $stepg0 = 1
```

5 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® Fortran コンパイラー・プロフェッショナル・エディションに同梱されているインテル® マス・カーネル・ライブラリー (インテル® MKL) の変更点、新機能、および最新情報をまとめています。

5.1 本バージョンでの変更

5.1.1 新機能

- LAPACK 3.2
  - 238 個の新しい LAPACK 関数
  - 超精密反復法の改良
  - ハウスホルダー QR 因数分解の非負対角
  - 低プロファイル行列でのハイパフォーマンス QR とハウスホルダー反射
  - 高速で正確な新しいヤコビ法 SVD
  - 矩形フル圧縮形式のルーチン
  - ピボットコレスキー
  - 混合精度反復法の改良 (コレスキー)
  - より安定した DQDS アルゴリズム
- DZGEMM 拡張 BLAS 関数の実装 (http://www.netlib.org/blas/blast-forum/ の説明を参照)。リファレンス・マニュアルの BLAS セクションの *gemm 関数ファミリーの説明を参照してください。
- PARDISO で実数、複素数、単精度データをサポート

5.1.2 ユーザービリティー/インターフェイスの向上

- スパース行列形式変換ルーチン:
  - CSR (3-配列バリエーション) ↔ CSC (3-配列バリエーション)
  - CSR (3-配列バリエーション) ↔ 対角形式
  - CSR (3-配列バリエーション) ↔ スカイライン
- Fortran95 BLAS と LAPACK のコンパイル・モジュール・ファイル (.mod) が含まれています。
  - モジュールは、インテル® Fortran コンパイラーで事前にビルドされており、インクルード・ディレクトリーにあります (フルパス情報については、インテル® MKL ユーザーズ・ガイドを参照してください)。
  - ほかのコンパイラー用のソースも提供されています。
  - インターフェイスについてのドキュメントは、インテル® MKL ユーザーズ・ガイドを参照してください。
- FFTW3 インターフェイスが直接メイン・ライブラリーに統合されました。
  - デフォルトのインテル® Fortran コンパイラー規則と名前修飾で互換性のないコンパイラーでラッパーを作成するためのソースコードも提供されています。
  - 詳細は、リファレンス・マニュアルの付録 G を参照してください。
- DFTI_DESCRIPTOR_HANDLE が型の名前を表すようになりました。ユーザープログラムで型として参照できます。
- 最適化ソルバードメインのヤコビ行列計算ルーチンにパラメーターが追加され、ユーザーデータにアクセスできるようになりました (詳細は、リファレンス・マニュアルの djacobix 関数の説明を参照してください)。
- 64 ビット・アーキテクチャーでインテル® MKL の単精度 BLAS 関数 (頭文字 "s" または "c" の関数) から 64 ビット浮動小数点精度関数へのインターフェイス・マッピング呼び出しが追加されました (詳細は、インテル® MKL ユーザーズ・ガイドの「sp2dp」を参照してください)。
- 互換ライブラリー (「ダミーライブラリー」) が削除されました。

5.1.3 パフォーマンスの向上

- インテル® 64 アーキテクチャー用にさらにスレッド化された BLAS レベル 1、2 関数
  - レベル 1 関数 (ベクトル-ベクトル): (CS,ZD,S,D)ROT、(C,Z,S,D)COPY、(C,Z,S,D)SWAP

        - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.7-4.7 倍向上
        - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 14-130 倍向上
    - レベル 2 関数 (行列-ベクトル): (C,Z,S,D)TRMV、(S,D)SYMV、(S,D)SYR、(S,D)SYR2
        - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.9-2.9 倍向上
        - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 16-40 倍向上
- インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー (5300 番台、5400 番台、7400 番台) で、DSYRK の 32 ビット逐次バージョンに導入された再帰アルゴリズムのパフォーマンスが最大 20% 向上。
- インテル® Xeon® 7460 プロセッサーで、大規模な問題の LU 因数分解 (DGETRF) がバージョン 10.1 Update 1 に対して 25% 向上。また小規模な問題でも劇的に向上。
- BLAS *TBMV/*TBSV 関数でレベル 1 BLAS 関数を使用。インテル® Core™ i7 プロセッサー上で 最大 3%、インテル® Core™2 プロセッサー 5300 番台と 5400 番台で最大 10% のパフォーマンスが向上。
- DGEMM パフォーマンスを強化するスレッド化アルゴリズムの向上
    - 8 スレッドで最大 7% の向上、3、5、7 スレッドで最大 50% の向上 (インテル® Core™ i7 プロセッサー)
    - 3 スレッドで最大 50% の向上 (インテル® Xeon® プロセッサー 7400 番台)
- 非素数サイズのスレッド化 1D 複素数-複素数 FFT
- 3D 複素数-複素数変換の新しいアルゴリズムにより 1 スレッドまたは 2 スレッドで小さな問題サイズ (最大 64x64x64) についてより優れたパフォーマンスを提供
- 対称正定行列の演算時におけるアウトオブコア (OOC) PARDISO のハイレベルな並列化実装
- すべての行列の型でインコアとアウトオブコアの両方で PARDISO のメモリー使用量が減少
- 実対称行列、複素エルミート行列、複素対称行列に対し PARDISO OOC で使用されるメモリーがインテル® MKL 10.1 で使用されていたメモリーの半分以下まで減少
- PARDISO/DSS における順序付けの並列化とシンボリック因子分解
- インテル® Core® i7 プロセッサーとインテル® Core™2 プロセッサーで次の VML 関数において最大 2 倍のパフォーマンスの向上 (平均で 30% の向上): v(s,d)Round、v(s,d)Inv、v(s,d)Div、v(s,d)Sqrt、v(s,d)Exp、v(s,d)Ln、v(s,d)Atan、v(s,d)Atan2
- インテル® Advanced Vector Extension (インテル® AVX) で次の関数の最適化バージョンが利用可能
    - BLAS: DGEMM
    - FFT
    - VML: exp、log、pow
    - 上記の関数にアクセスする `mkl_enable_instructions()` 関数に関する重要な情報については、インテル® MKL ユーザーズ・ガイドを参照してください。

5.2 既知の問題

本リリースにおける既知の制限事項の詳細なリストは、http://software.intel.com/en-us/articles/intel-math-kernel-library-support-resources/ (英語) を参照してください。

5.3 注意事項

インテル® MKL の将来のバージョンでは以下の変更が予定されています。「テクニカルサポート」を参照してください。

- ファイル名に `solver` を含むライブラリーの内容をコア・ライブラリーに移動する予定です。これらの `solver` ライブラリーはその後削除される予定です。

5.4 権利の帰属

エンド・ユーザー・ソフトウェア使用許諾契約書 (End User License Agreement) で言及されているように、製品のドキュメントおよび Web サイトの両方で完全なインテル製品名の表示 (例えば、"インテル® マス・カーネル・ライブラリー") とインテル® MKL ホームページ (www.intel.com/software/products/mkl (英語)) へのリンク/URL の提供を正確に行うことが最低限必要です。

インテル® MKL の一部の基となった BLAS の原版は http://www.netlib.org/blas/index.html (英語) から、

LAPACK の原版は http://www.netlib.org/lapack/index.html (英語) から入手できます。LAPACK の開発は、E. Anderson、Z. Bai、C. Bischof、S. Blackford、J. Demmel、J. Dongarra、J. Du Croz、A. Greenbaum、S. Hammarling、A. McKenney、D. Sorensen らによって行われました。LAPACK 用 FORTRAN 90/95 インターフェイスは、http://www.netlib.org/lapack95/index.html (英語) にある LAPACK95 パッケージと類似しています。すべてのインターフェイスは、純粋なプロシージャー用に提供されています。

インテル® MKL クラスター・エディションの一部の基となった ScaLAPACK の原版は http://www.netlib.org/scalapack/index.html (英語) から入手できます。ScaLAPACK の開発は、L. S. Blackford、J. Choi、A. Cleary、E. D'Azevedo、J. Demmel、I. Dhillon、J. Dongarra、S. Hammarling、G. Henry、A. Petitet、K. Stanley、D. Walker、R. C. Whaley らによって行われました。

インテル® MKL の PARDISO は、バーゼル大学 (University of Basel) から無償で提供されている PARDISO 3.2 (http://www.pardiso-project.org (英語)) と互換性があります。

本リリースのインテル® MKL の一部の FFT 関数は、カーネギーメロン大学からライセンスを受けて、SPIRAL ソフトウェア生成システム (http://www.spiral.net/ (英語)) によって生成されました。本リリースのインテル® MKL の一部の FFT 関数は、ヒューストン大学からライセンスを受けて、UHFFT ソフトウェア生成システムによって生成されました。SPIRAL の開発は、Markus Püschel、José Moura、Jeremy Johnson、David Padua、Manuela Veloso、Bryan Singer、Jianxin Xiong、Franz Franchetti、Aca Gacic、Yevgen Voronenko、Kang Chen、Robert W. Johnson、Nick Rizzolo らによって行われました。

6　著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む)に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本書で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、インテルの Web サイトを参照してください。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

© 2009 Intel Corporation. 無断での引用、転載を禁じます。