Intel® Math Kernel Library Version 11.1 （Windows 版）

～ 活用ガイド～

XLSOFT

2014年 8月 8日 作成版

インテル® MKL について

- 最適化
  - 各世代のインテル® プロセッサーがサポートする拡張命令セット（SSE またはインテル® AVX など）の使用、キャッシュメモリの活用など
  - 実行環境に応じて、適切なコードを選択（自動ディスパッチャー）
- 並列化
  - OpenMP による共有メモリ環境での並列化（マルチスレッド）
  - MPIによる分散メモリ環境での並列化（マルチプロセス）
- C/C++ 言語および Fortran 言語から利用可能
- 動的リンクまたは静的リンクを選択可能
- シングルスレッド／マルチスレッドバージョンを選択可能
- 実行、および再配布におけるロイヤリティーフリー
  （詳細や制限については、使用許諾をご確認ください）

利用可能な関数(演算処理機能)

- BLAS, Sparse BLAS(基本線形代数演算)
- LAPACK(連立1次方程式、固有値および最小二乗問題)
- スパース・ソルバー
  - PARDISO および DSS インターフェイスルーチン(直接法ソルバー)
  - RCI(Reverse Communication Interface)による反復法ソルバー
- 数学関数:VML(Vector Math Library)
- 統計関数:VSL(Vector Statistical Library)
  - RNG(乱数発生器)、相関および畳み込み演算
  - サマリー統計ライブラリー
- FFT(高速フーリエ変換)※ FFTW インターフェイスを含む
- 偏微分方程式(PDE)のサポート
  - TT(sin/cos 変換など), Laplace / Poisson / Helmholtz ソルバー
- 非線形最適化問題(TR:信頼領域法)
- データフィッティング(スプライン補間)
- クラスター対応ルーチン:PBLAS, ScaLAPACK, BLACS, クラスター FFT

MKL 開発環境

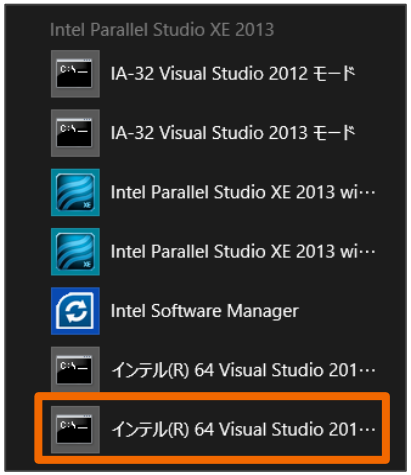
	Windows*	Linux*	OS X*
コンパイラー	Intel、CVF、Microsoft	Intel、GNU	Intel、GNU
ライブラリー	.lib .dll	.a .so	.a .dylib

ドメイン	Fortran 77	Fortran 90/95	C/C++
BLAS	✔	✔	CBLAS 経由
Sparse BLAS Level 1	✔	✔	CBLAS 経由
Sparse BLAS Level 2&3	✔	✔	✔
LAPACK	✔	✔	✔
ScaLAPACK	✔		
PARDISO	✔	✔	✔
DSS & ISS	✔	✔	✔
VML / VSL	✔	✔	✔
FFT / Cluster FFT		✔	✔
PDEs		✔	✔
Optimization (TR) Solvers	✔	✔	✔
SSL	✔	✔	✔

ビルド環境（コマンドライン）

◆ Windows の [スタート] メニューからインテル® コンパイラー専用コマンドプロンプトを起動（必要な環境変数が自動で設定される）

Windows 8



例）「インテル® 64 Visual Studio 2013 モード」を選択した場合

¥Composer XE 2013 SP1¥bin ¥ipsxe-comp-vars.bat intel64

↓（スクリプトコール）

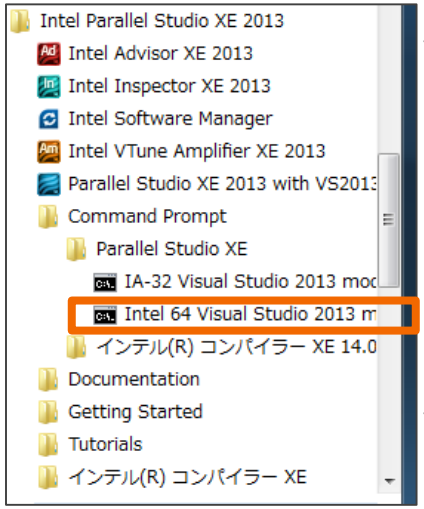
¥Inspector XE 2013¥ inspxe-vars.bat

¥VTune Amplifier XE 2013¥ amplxe-vars.bat

¥compilervars.bat intel64 vs2013

↓（スクリプトコール）

Windows 7



¥tbb¥bin¥tbbvars.bat intel64 vs2013

¥mkl¥bin¥mklvars.bat intel64 vs2013

MKLROOT=¥Composer XE 2013 SP1¥mkl

PATH=¥Composer XE 2013 SP1¥redist¥intel64¥mkl
¥Composer XE 2013 SP1¥redist¥intel64¥compiler

LIB= ¥Intel¥Composer XE 2013 SP1¥mkl¥lib¥intel64
¥Intel¥Composer XE 2013 SP1¥compiler¥lib¥intel64

INCLUDE= ¥Intel¥Composer XE 2013 SP1¥mkl¥include

¥ipp¥bin¥ippvars.bat intel64 vs2013

¥compilervars_arch.bat intel64 vs2013

Fortran 95 モジュールの環境設定

◆ 以下の環境変数設定ファイルに mod 引数を指定して手動で環境設定する。

<インストールディレクトリ>¥mkl¥bin¥mklvars.bat

- mklvars.bat ia32 mod

INCLUDE= ¥Intel¥Composer XE 2013 SP1¥mkl¥include
¥Intel¥Composer XE 2013 SP1¥mkl¥include¥ia32

- mklvars.bat intel64 mod lp64

INCLUDE= ¥Intel¥Composer XE 2013 SP1¥mkl¥include
¥Intel¥Composer XE 2013 SP1¥mkl¥include¥intel64¥lp64

- mklvars.bat intel64 mod ilp64

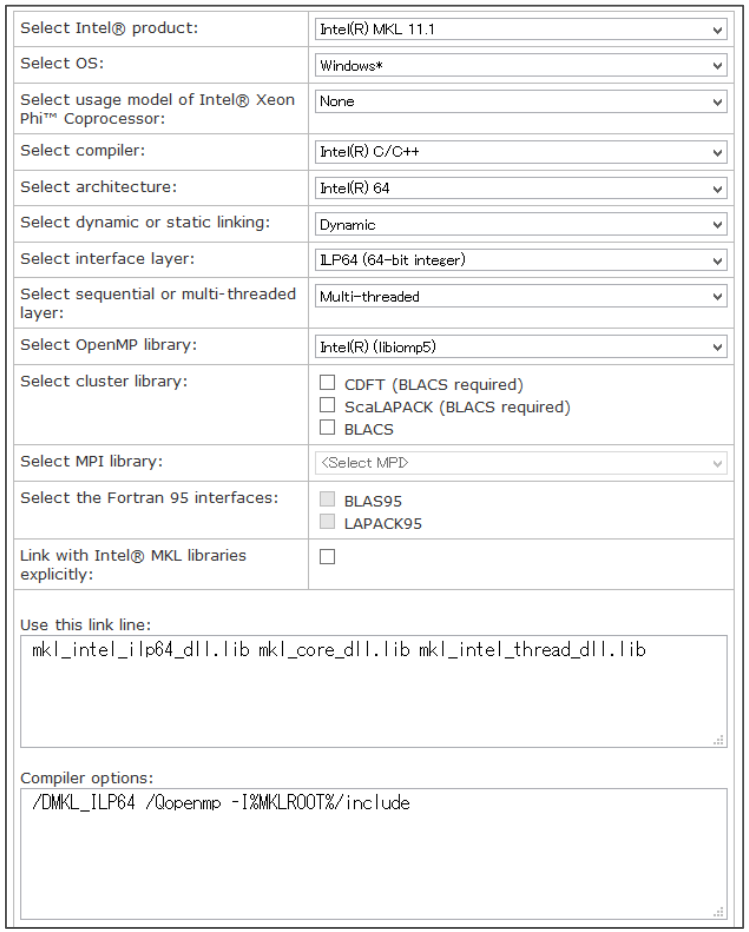
INCLUDE= ¥Intel¥Composer XE 2013 SP1¥mkl¥include
¥Intel¥Composer XE 2013 SP1¥mkl¥include¥intel64¥ilp64

<インストールディレクトリ>(デフォルト)

C:¥Program Files (x86)¥Intel (x64システム)
C:¥Program Files¥Intel (x86システム)

リンク・アドバイザー・ツールのご利用

http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor
または、
<インストール・フォルダー>¥Documentation¥en_US¥mkl¥mkl_link_line_advisor.htm



① 各種項目を選択

◆ 項目内容
- バージョン
- OS
- コプロセッサーの利用
- 使用コンパイラー
- アーキテクチャー
- リンク方法
- Integer のビット幅
- シングル/マルチスレッド
- OpenMPライブラリーの使用
- クラスターライブラリーの使用
- MPIの選択
- Fortranでの使用設定
- 動的リンクライブラリーの明記

② 必要なリンク情報とコンパイルオプションが表示される。

リンクモデル

- 静的リンク
- 動的リンク
- カスタム動的リンク
- 動的リンク(Single Dynamic Library (SDL) interface の利用)

	静的リンク	動的リンク	カスタム動的リンク	動的リンク(SDL)
CPU ディスパッチ	自動	自動	再コンパイル	自動
最適化	全 CPU	全 CPU	全 CPU	全 CPU
ビルド方法	静的リンク・ライブラリーのリンク	インポート・ライブラリーのリンク	カスタムビルドしたインポート・ライブラリーのリンク	"mkl_rt.lib" ライブラリーのリンク
呼出関数名	通常の関数名	通常の関数名	通常の関数名	通常の関数名
トータル・バイナリー・サイズ	小さい	大きい	小さい	非常に大きい
アプリケーション・サイズ	小さい	非常に小さい	非常に小さい	非常に小さい
マルチスレッド	可能	可能	可能	可能

リンク・レイヤー・モデル

① インターフェイス・レイヤー
- 呼出規約(cdecl / stdcall)
- Integer タイプ(LP64 / ILP64)

② スレッドレイヤー
- シーケンシャル・バージョン(Sequential)
- マルチスレッド・バージョン(Thread)

③ 計算レイヤー
- 計算コア・ライブラリー

④ ランタイム・レイヤー
- インテル提供 OpenMP ランタイム・ライブラリー

例)Windows	インターフェイス(.lib)	スレッド(.lib)	計算(.lib)	ランタイム(.lib)
IA-32 静的リンク	mkl_intel_c	mkl_sequential	mkl_core	
IA-32 動的リンク	mkl_intel_c_dll	mkl_intel_thread_dll	mkl_core_dll	libiomp5md
Intel64 静的リンク	mkl_intel_lp64	mkl_intel_thread	mkl_core	libiomp5md
Intel64 動的リンク	mkl_intel_ilp64_dll	mkl_sequential_dll	mkl_core_dll	
IA-32 / Intel64 動的リンク(SDL)	mkl_rt			

代表的な MKL ライブラリー

IA-32／静的	IA-32／動的	Intel64 ／静的	Intel64 ／動的
◆ インターフェイス・レイヤー			
mkl_intel_c.lib（cdecl）	mkl_intel_c_dll.lib（cdecl）	mkl_intel_lp64.lib（LP64）	mkl_intel_lp64_dll.lib（LP64）
mkl_intel_s.lib（stdcall）	mkl_intel_s_dll.lib（stdcall）	mkl_intel_ilp64.lib（ILP64）	mkl_intel_ilp64_dll.lib（ILP64）
	mkl_rt.lib（SDL）		mkl_rt.lib（SDL）
◆ スレッドレイヤー			
mkl_intel_thread.lib（マルチスレッド）	mkl_intel_thread_dll.lib（マルチスレッド）	mkl_intel_thread.lib（マルチスレッド）	mkl_intel_thread_dll.lib（マルチスレッド）
mkl_sequential.lib（シングルスレッド）	mkl_sequential_dll.lib（シングルスレッド）	mkl_sequential.lib（シングルスレッド）	mkl_sequential_dll.lib（シングルスレッド）
◆ 計算レイヤー			
mkl_core.lib	mkl_core_dll.lib	mkl_core.lib	mkl_core_dll.lib
◆ RTL レイヤー			
libiomp5md.lib	libiomp5md.lib	libiomp5md.lib	libiomp5md.lib

※ 詳細は、製品の MKL ユーザーズガイドの「付録 C: 詳細なディレクトリー構造」を参照

コマンド例

例1）Windows／IA-32 対応インテル® コンパイラー／静的リンク／シングルスレッド

```
> icl myprog.c   mkl_intel_c.lib mkl_sequential.lib mkl_core.lib
```

```
> ifort myprog.f mkl_intel_c.lib mkl_sequential.lib mkl_core.lib
```

例2）Windows／Intel64 対応インテル® コンパイラー／動的リンク／マルチスレッド

```
> icl myprog.c   mkl_intel_lp64_dll.lib mkl_intel_thread_dll.lib
mkl_core_dll.lib libiomp5md.lib
```

```
> ifort myprog.f mkl_intel_lp64_dll.lib mkl_intel_thread_dll.lib
mkl_core_dll.lib libiomp5md.lib
```

例3）Windows／インテル® コンパイラー／動的リンク（SDL利用）

```
> icc myprog.c   mkl_rt.lib
```

```
> ifort myprog.f mkl_rt.lib
```

例4）Windows／インテル® コンパイラー／静的リンク／マルチスレッド

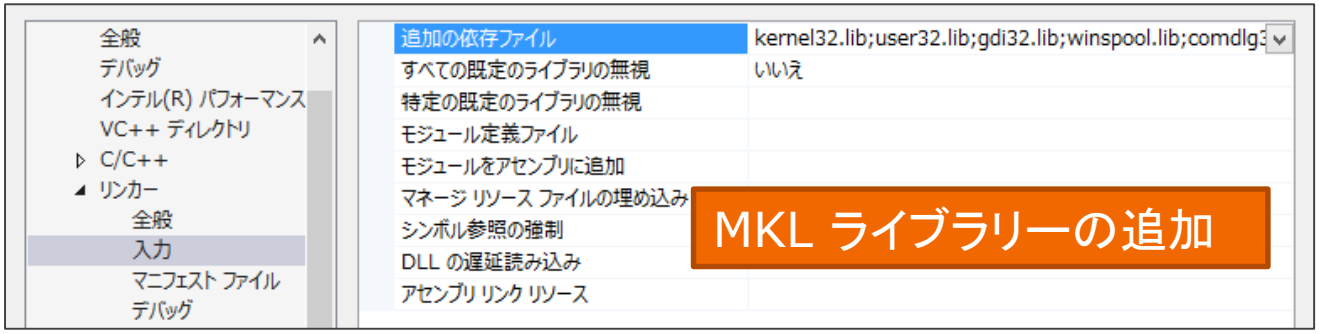
```
> icc myprog.c   /Qmkl:parallel /MT
```

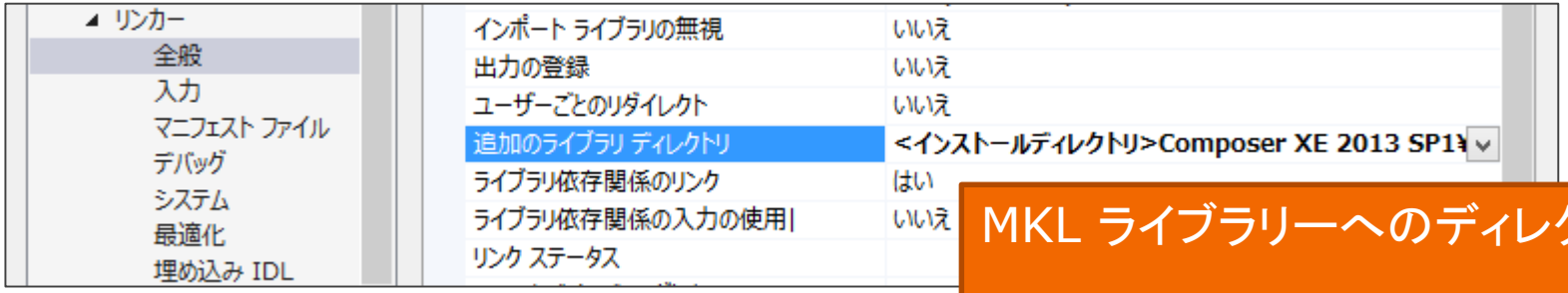
```
> ifort myprog.f /Qmkl:parallel /MT
```

Visual Studio からのリンク(Visual C++ 使用)

XLSOFT

◆ プロジェクトの [プロパティページ] からそれぞれの内容を設定する。

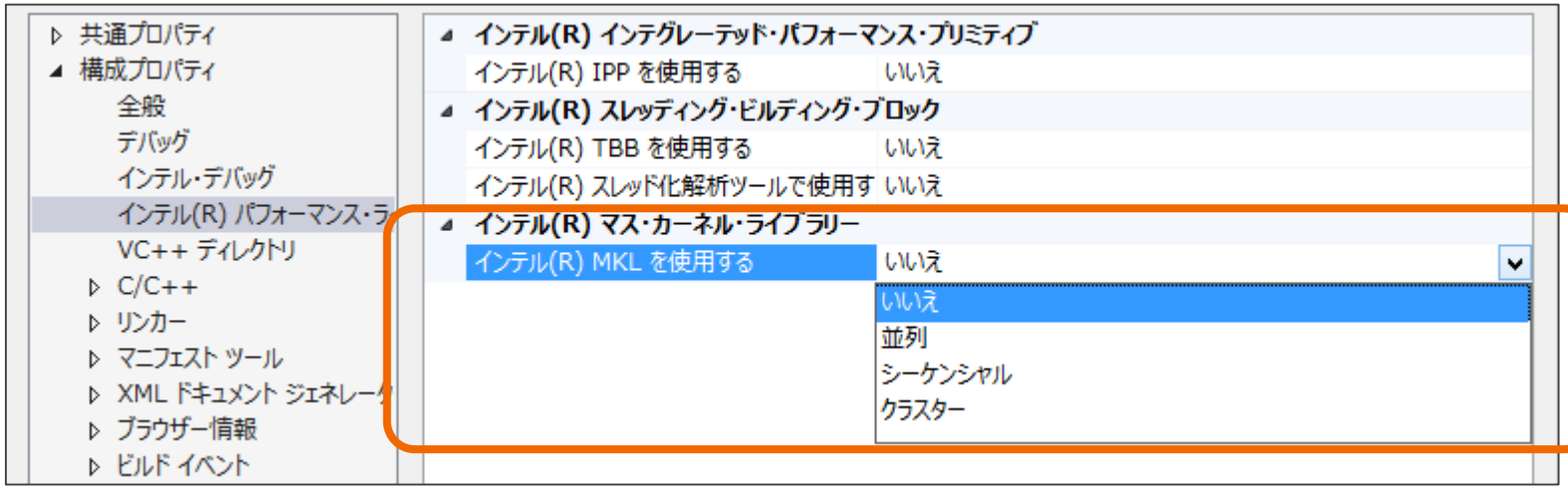




Visual Studio からのリンク(Intel Compiler 使用)

XLSOFT

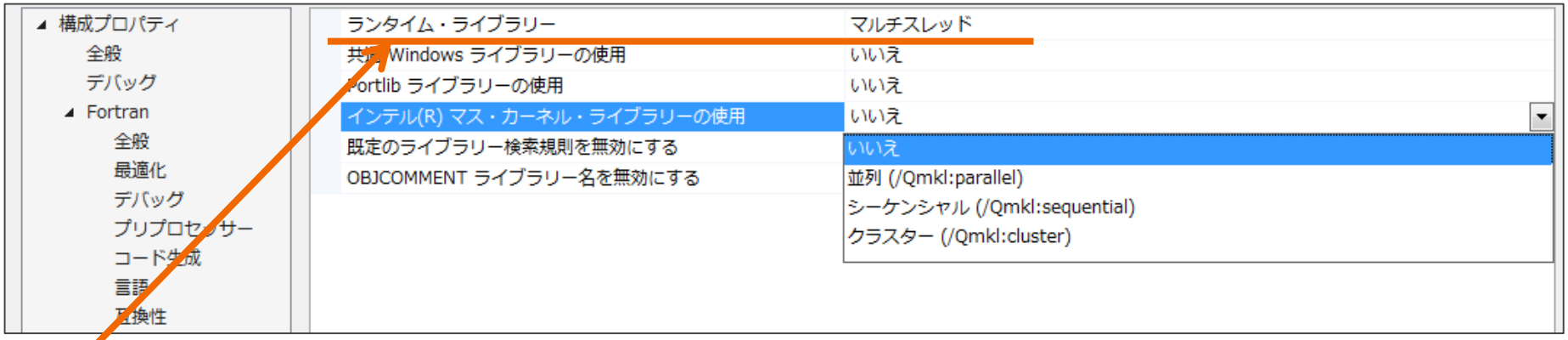
Visual Studio 2013 でインテル® C++ Composer XE 製品をご使用の場合は、プロジェクトの [プロパティ ページ] の“インテル(R) パフォーマンス・ライブラリー”から MKL のビルド環境設定が可能。



静的／動的リンクの指定は、プロジェクトの [プロパティ ページ] の [C/C++] - [コード生成] - [ランタイム ライブラリ] の設定内容(/MT、/MD など)で決定される。

VSからのリンク(インテル® Visual Fortran Composer XE)

インテル® Visual Fortran Composer XE 製品をご使用の場合は、プロジェクトの[プロパティ ページ] の“インテル(R) マス・カーネル・ライブラリーの使用”からMKL のビルド環境設定が可能。



静的／動的リンクの指定は、プロジェクトの [プロパティ ページ] の[Fortran] - [ライブラリー] - [ランタイム・ライブラリー] の設定内容で決定される。

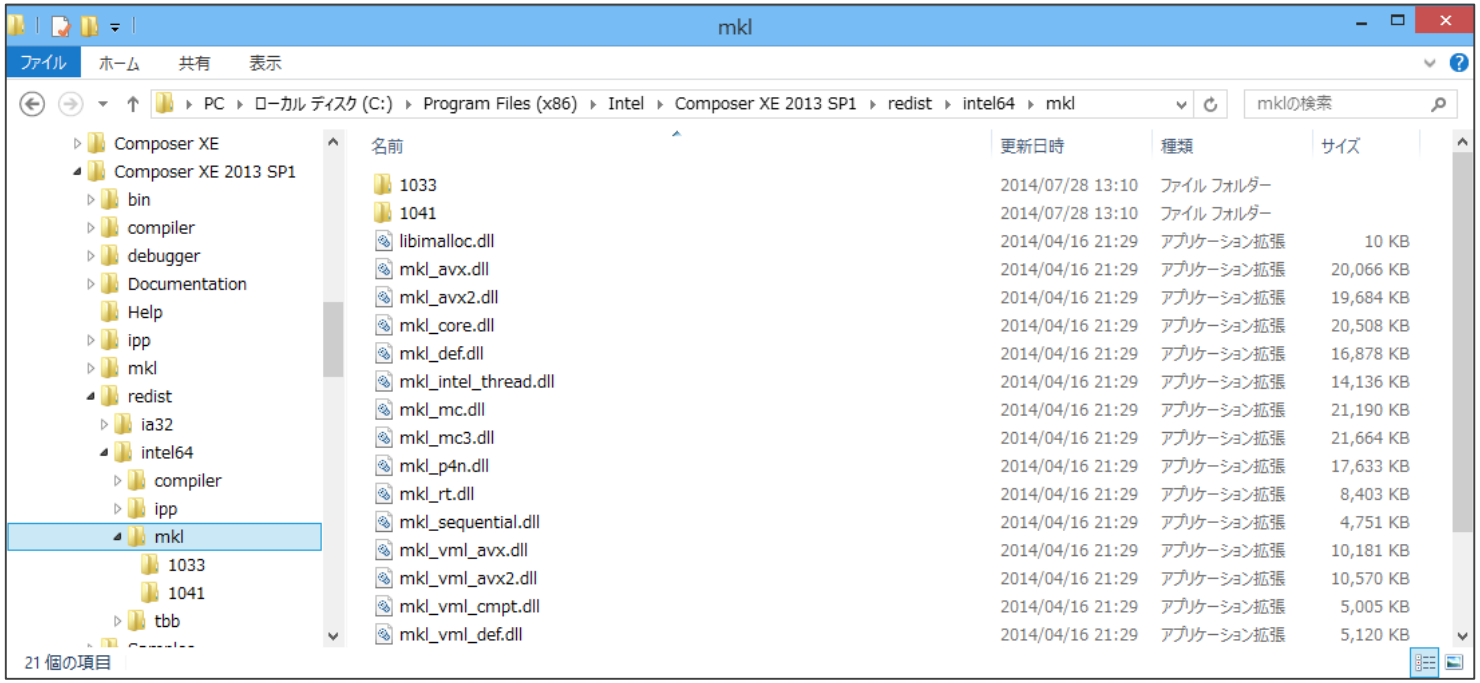
MKL ランタイム・ライブラリー

■ 実行環境で必要となる MKL ランタイム・ライブラリー

<インストール・ディレクトリー>¥redist¥ia32¥mkl¥*.dll

<インストール・ディレクトリー>¥redist¥intel64¥mkl¥*.dll

例）C:¥Program Files (x86)¥Intel¥Composer XE 2013 SP1¥redist¥intel64¥mkl



※再配布可能ファイル一覧は以下のファイルを参照
<インストール・ディレクトリー>¥Documentation¥ja_JP¥mkl¥redist.txt

MKL ランタイム・ライブラリー（続き）

- インテル OpenMP ランタイム・ライブラリー（libiomp5md.dll）

**マルチスレッド（並列）用の MKL ライブラリーをリンクした場合は、
実行環境に libiomp5md.dll ランタイム・ライブラリーが必要となります。**

<インストール・ディレクトリー>¥redist¥ia32¥compiler¥libiomp5md.dll

<インストール・ディレクトリー>¥redist¥intel64¥compiler¥libiomp5md.dll

※または以下のURL からランタイム環境構築のパッケージをインストールします。
http://software.intel.com/en-us/articles/redistributable-libraries-for-the-intel-c-and-visual-fortran-composer-xe-for-windows/

マルチスレッド対応

XLSOFT

- 多数の MKL 関数がマルチスレッド化されている
  詳細は製品の MKL ユーザーズガイドの [パフォーマンスとメモリーの管理] - [並列処理の使用] - [スレッド化される関数とスレッド化される問題] を参照

- MKL のマルチスレッドは、OpenMP で実装されている。

- MKL は、スレッドセーフに作成されているので、複数のスレッドから同時に呼び出し、安全に使用することができる。

- MKL 関数は、並列実行領域内からコールされる場合、それを検知してシングルスレッドで動作する。

- マルチスレッド MKL 関数を使用する場合は、プログラムを /MT オプションでビルドしたほうが効果的。
  MKL_DYNAMIC=TRUE がデフォルト設定で、スレッド数は物理コア数となる。
  また、MKL_DYNAMIC=FALSE とした場合でも、データ量が少ない場合は、マルチスレッドしない場合がある。

- mkl_set_dynamic() 関数でも設定可能。

スレッド数の制御方法

環境変数:
- OMP_NUM_THREADS
- MKL_NUM_THREADS
- MKL_DOMAIN_NUM_THREADS

関数:
- omp_set_num_threads() ;
- mkl_set_num_threads() ;
- mkl_domain_set_num_threads() ;

※環境変数を使用する場合は、プログラムの実行中で設定を変更することはできない。

優先順位の基準:

MKL > OpenMP

mkl_domain_set_... > mkl_set_...

関数 > 環境変数

1. mkl_domain_set_num_threads();
2. MKL_DOMAIN_NUM_THREADS
3. mkl_set_num_threads();
4. MKL_NUM_THREADS
5. omp_set_num_threads();
6. OMP_NUM_THREADS

設定例:(Windows)

```
mkl_domain_set_num_threads ( 2, MKL_BLAS );
MKL_DOMAIN_NUM_THREADS="MKL_ALL=1, MKL_BLAS=4"
mkl_set_num_threads(2);
MKL_NUM_THREADS=4
omp_set_num_threads(2);
OMP_NUM_THREADS=4
```

パフォーマンス Tips

◆ データ・アライメントは16バイト境界が効果的
mkl_malloc 関数が利用できる。
例）
メモリー取得:`mkl_malloc( sizeof(double)*workspace, 16 );`
メモリー解放:`mkl_free( darray );`

◆ HT テクノロジーの無効化
・スレッド間で同様な計算が実施されるので HT テクノロジーを有効利用できない可能性が高い。

◆ アフィニティー・マスクの利用
・スレッドを特定のコアに常に割り当てたほうが効果的
例）set KMP_AFFINITY=granularity=fine,compact,1,0

◆ /FTZ /DAZ の有効化
・デノーマル浮動小数点演算の回避

MKL メモリー管理機能

XLSOFT

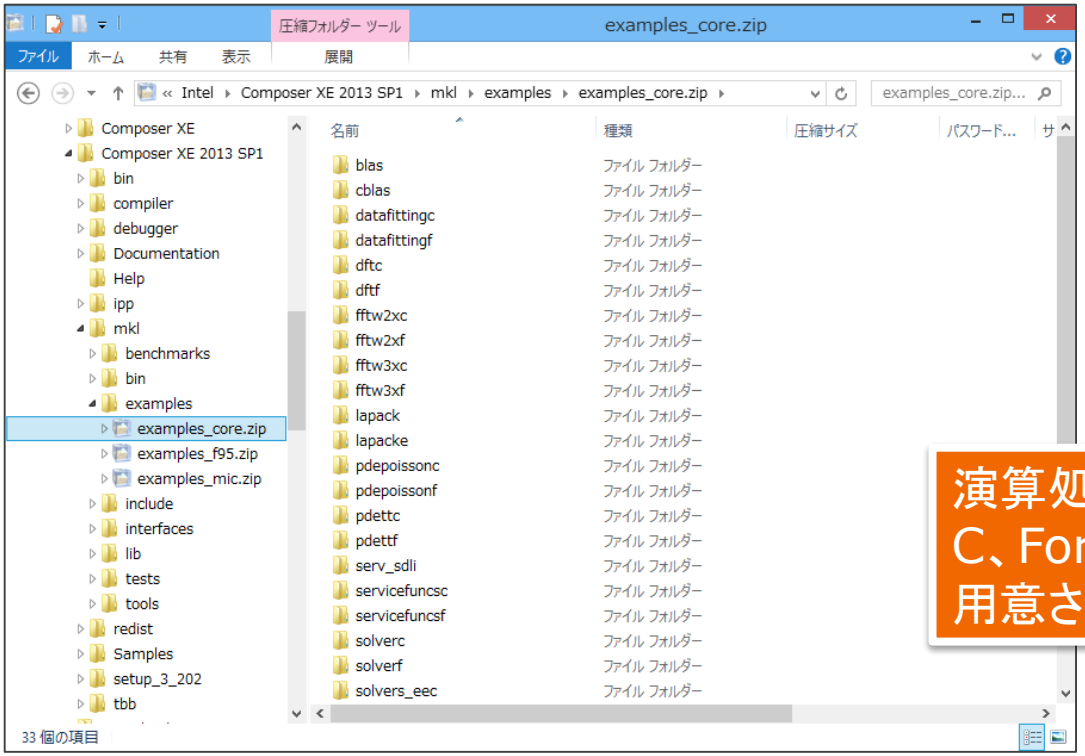
◆ MKL 関数によって暗黙的に取得、利用されるメモリー領域は、再利用の目的で、アプリケーションが終了するまで解放されない。

→ このために、メモリーリークと判断される場合がある。

- mkl_mem_stat() 関数で、そのメモリー量を取得できる。
- mkl_free_buffers() 関数で、そのメモリー領域を解放できる。
- MKL_DISABLE_FAST_MM 環境変数、または、mkl_disable_fast_mm() 関数で、メモリー管理機能を無効化することができる。

MKL サンプルコード

サンプルコード配置場所
<インストール・ディレクトリー>¥mkl¥examples¥



演算処理機能ごとに
C、Fortran 言語の2種類が
用意されている

◆ ドメイン／言語別の MKL 関数の利用方法が分かる。
◆ 各フォルダーに含まれる Make ファイルより、リンクライブラリーが分かる。