Intel® C++ Composer XE 2011 for Mac OS* X Installation Guide and Release Notes

Document number: 321413-003US
4 August 2011

Table of Contents

1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and includes notes about features and problems not described in the product documentation.

Intel® C++ Composer XE 2011 is the next release of the product formerly called Intel® C++ Compiler Professional Edition.

1.1 Change History

This section highlights important changes in product updates.

Update 6 (2011.6)

- Mac OS* X 10.5 is no longer supported
- The product installs into a new top-level folder
- Intel® C++ Compiler XE 12.1
  - Additional C++0x features supported
  - Additional compiler options
  - Full Intel® Cilk™ Plus support
  - Change to Cilk Plus array notation semantics
  - Enhancements to OpenMP* support
  - The core compiler documentation known as the User and Reference Guides has been reorganized and streamlined. Among the most noticeable changes are: a new Key Features section highlighting important Intel compiler functionality and the organization of the Compiler Option reference section into functional groups.
- Intel® Debugger 12.1
- Intel® Math Kernel Library updated to 10.3 Update 6

- Intel® Integrated Performance Primitives 7.0 Update 5
- Intel® Threading Building Blocks 4.0
- Corrections to reported problems

Update 5 (2011.5)

- Intel® Math Kernel Library updated to 10.3 Update 5
- Intel® Threading Building Blocks 3.0 Update 8
- Corrections to reported problems

Update 4 (2011.4)

- Intel® Math Kernel Library updated to 10.3 Update 4
- Intel® Integrated Performance Primitives 7.0 Update 4
- Intel® Threading Building Blocks 3.0 Update 7
- -export and –export-dir deprecated
- Corrections to reported problems

Update 3 (2011.3)

- Intel® Math Kernel Library updated to 10.3 Update 3
- Intel® Integrated Performance Primitives 7.0 Update 3
- Intel® Threading Building Blocks 3.0 Update 6
- `-sox` option enhancement
- Deprecating Mac OS* X 10.5.8 support
- Corrections to reported problems

Update 2 (2011.2)

- Intel® Math Kernel Library updated to 10.3 Update 2
- Intel® Integrated Performance Primitives 7.0 Update 2
- Intel® Threading Building Blocks 3.0 Update 5
- 3 intrinsics changed in immintrin.h
- Utility “inspxe-runsc.exe” changed
- Corrections to reported problems

Update 1 (2011.1)

- Intel® Math Kernel Library updated to 10.3 Update 1
- Corrections to reported problems

Product Release (2011.0)

- Initial product release

1.2 Product Contents

Intel® C++ Composer XE 2011 Update 6 for Mac OS X* includes the following components:

- Intel® C++ Compiler XE 12.1 for building applications that run on Intel-based Mac systems running the Mac OS* X operating system
- Intel® Debugger 12.1
- Intel® Integrated Performance Primitives 7.0 Update 5
- Intel® Math Kernel Library 10.3 Update 6
- Intel® Threading Building Blocks 4.0
- Integration into the Xcode* development environment
- On-disk documentation

1.3 System Requirements

- An Intel®-based Apple* Mac* system
- 1GB RAM minimum, 2GB RAM recommended
- 3GB free disk space
- One of the following combinations of Mac OS* X, Xcode* and the Xcode SDK:
  - OS X 10.7 and Xcode 4.1 and SDK 10.7
  - OS X 10.6.8 and Xcode 4.0 and SDK 10.6
  - OS X 10.6.8 and Xcode 3.2.5 and SDK 10.6
- gcc* 4

Note: Advanced optimization options or very large programs may require additional resources such as memory or disk space.

1.4 Documentation

Product documentation can be found in the `Documentation` folder as shown under Installation Folders.

Optimization Notice

Intel compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel Compiler User and Reference Guides” under “Compiler Options." Many library routines that are part of Intel compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel compilers, associated libraries and associated development tools may or may not optimize

to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20110307

1.5 Technical Support

If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit:
http://www.intel.com/software/products/support/

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

2 Installation

The installation of the product requires a valid license file or serial number. If you are evaluating the product, you can also choose the “Evaluate this product (no serial number required)” option during installation.

If you will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel C++ Compiler afterwards.

You will need to have administrative or “sudo” privileges to install, change or uninstall the product.

If you received the compiler product on DVD, insert the DVD. Locate the disk image file (`xxx.dmg`) on the DVD and double-click on it. If you received the compiler product as a download, double-click the downloaded file.

Follow the prompts to complete installation.

Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions.

2.1 Using a License or Serial Number from Intel® C++ Compiler 11.1 Professional Edition to Install

Serial numbers and licenses distributed for use with the Intel® C++ Compiler 11.1 Professional Edition will not work with Intel® C++ Composer XE 2011. You can obtain a new upgraded license and serial number for free if your current product is active by doing the following:

1. Login to the Intel® Registration Center by entering your Login ID and Password in the Registered Users Login section of the web page. You will find a list of all products you have subscriptions for in the "My Products" page.
2. For the current product, you will see the XE product name displayed in addition to the original product name. Clicking the latest update in the download latest update column leads you to the product upgrade page. Click the product name to upgrade.
3. You can now send yourself an email with the updated license file or use the updated serial number to install the C++ Composer XE 2011 product.

2.2 Activation of Purchase after Evaluation Using the Intel Activation Tool

Note for evaluation customers a new tool Intel Activation Tool "`ActivationTool`" is included in this product release and installed at `/opt/intel/composerxe-2011.x.xxx/Activation` directory.

If you installed the product using an Evaluation license or SN, or using the "Evaluate this product (no serial number required)" option during installation, and then purchased the product, you can activate your purchase using the Intel Activation Tool at `/opt/intel/composerxe-2011.x.xxx/Activation/ActivationTool`. It will convert your evaluation software to a fully licensed product. To use the tool:

```
$ /opt/intel/composerxe-2011.x.xxx/Activation/ActivationTool
[SN_Num_here]
```

2.3 Using a License Server

If you have purchased a "floating" license, see http://software.intel.com/en-us/articles/licensing-setting-up-the-client-floating-license/ for information on how to install using a license file or

license server. This article also provides a source for the Intel® License Server that can be installed on any of a wide variety of systems.

2.4 Installation Folders

The compiler installs, by default, under `/opt/intel` – this is referenced as `<install-dir>` in the remainder of this document. You are able to specify a different location. If Xcode integration is installed, a second copy of these files is present under `/Developer/opt/intel`.

The directory organization has changed since the Intel® Compilers 11.1 release.

While the top-level installation directory has also changed between the original C++ Composer XE 2011 release and Composer XE 2011 Update 6, the `composerxe` symbolic link can still be used to reference the latest product installation.

Under `<install-dir>` are the following directories:

- `bin` – contains symbolic links to executables for the latest installed version
- `lib` – symbolic link to the lib directory for the latest installed version
- `include` – symbolic link to the include directory for the latest installed version
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `ipp` – symbolic link to the directory for the latest installed version of Intel® Integrated Performance Primitives
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `tbb` – symbolic link to the directory for the latest installed version of Intel® Threading Building Blocks
- `composerxe` – symbolic link to the `composer_xe_2011_sp1` directory
- `composer_xe_2011_sp1` – directory containing symbolic links to subdirectories for the latest installed Intel® Composer XE 2011 compiler release
- `composer_xe_2011_sp1.<n>.<pkg>` - physical directory containing files for a specific compiler version. `<n>` is the update number, and `<pkg>` is a package build identifier.

Each `composer_xe_2011_sp1` directory contains the following directories that reference the latest installed Intel® Composer XE 2011 compiler:

- `bin` – directory containing scripts to establish the compiler environment and symbolic links to compiler executables for the host platform
- `pkg_bin` – symbolic link to the compiler `bin` directory
- `include` – symbolic link to the compiler `include` directory
- `lib` – symbolic link to the compiler `lib` directory
- `ipp` – symbolic link to the `ipp` directory
- `mkl` – symbolic link to the `mkl` directory
- `tbb` – symbolic link to the `tbb` directory

- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the `man` directory
- `Documentation` – symbolic link to the `Documentation` directory
- `Samples` – symbolic link to the `Samples` directory

Each `composer_xe_2011_sp1.<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Composer XE 2011 compiler:

- `bin` – all executables
- `compiler` – shared libraries and header files
- `debugger` – debugger files
- `Documentation` – documentation files
- `man` – symbolic link to the `man` directory
- `ipp` – Intel® Integrated Performance Primitives libraries and header files
- `mkl` – Intel® Math Kernel Library libraries and header files
- `tbb` – Intel® Threading Building Blocks libraries and header files
- `Samples` – Product samples and tutorial files

If you have both the Intel C++ and Intel Fortran compilers installed, they will share folders for a given version and update.

This directory layout allows you to choose whether you want the latest compiler, no matter which version, the latest update of the Intel® Composer XE 2011 compiler, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh [.csh]` script, which will always get the latest compiler installed. This layout should remain stable for future releases.

2.5 Installing Intel® Integrated Performance Primitives Cryptography Libraries

The Intel® Integrated Performance Primitives product provides an optional component containing libraries of cryptography routines. Installation and use of the cryptography libraries requires a separate license that is available at no charge from Intel once your license for Intel Integrated Performance Primitives has been registered. Export restrictions apply. For details, please see http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-cryptography-library/

2.6 Relocating Product After Install

The Xcode integration is relocatable simply by dragging and dropping the Xcode directory tree to another location. If you wish to use `idb` from a command prompt using a relocated Xcode directory tree, please see http://software.intel.com/en-us/articles/running-idb-from-command-line-after-relocating-xcode-environment/ for additional steps that are required. Note that `idb` is not available from within the Xcode IDE.

2.7 Removal/Uninstall

It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open Terminal and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command:
   `<install-dir>/composer_xe_2011_sp1.<n>.<pkg>/uninstall_cproc.sh`
3. Follow the prompts

If you are not currently logged in as `root` you will be asked for the `root` password.

3 Intel® C++ Compiler

This section summarizes changes, new features and late-breaking news about the Intel C++ Compiler.

3.1 New and Changed Features

C++ Composer XE 2011 Update 6 now contains Intel® C++ Compiler XE 12.1. The following features are new or significantly enhanced in this version. For more information on these features, please refer to the documentation.

- Intel® Cilk™ Plus task parallel language extensions (cilk_spawn, cilk_for, cilk_sync and reducers) now supported.
- Apple* Blocks
- Features from C++0x
  - Variadic templates
  - char16_t/char32_t types
  - Defaulted and deleted functions
  - sizeof, typeid or decltype can refer directly to a non-static data member of a class
  - nullptr
  - New-style SFINAE as described by N2634
  - Template aliases
  - Late-specified return types
  - Default template arguments for function templates
- OpenMP* 3.1
  - final clause
  - mergeable clause
  - taskyield directive
  - new atomic clauses
- Improved IDE integration for Static Security Analysis

The following features are new or significantly enhanced in Intel® C++ Compiler XE 12.0. For more information on these features, please refer to the documentation.

- Features from C++0x

    - `rvalue` references
    - Standard atomics
    - Support of C99 hexadecimal floating point constants when in “Windows C++” mode
    - Right angle brackets
    - Extended friend declarations
    - Mixed string literal concatenations
    - Support for `long long`
    - Variadic macros
    - Static assertions
    - Auto-typed variables
    - Extern templates
    - `__func__` predefined identifier
    - Declared type of an expression (decltype)
    - Universal character name literals
    - Strongly-typed enums
    - Lambdas
- An option to use math library functions that are faster but return results with less precision or accuracy
- An option to use math library functions that return consistent results across different models and manufacturers of processors

3.1.1 Intel® Cilk™ Plus fully supported in Update 6

Intel® Cilk™ Plus language extensions for implementing task parallelism which includes cilk_spawn, cilk_for, cilk_sync and Cilk Plus reducers are now supported on Mac OS* in Update 6. Please refer to the documentation for details.

3.1.2 Intel® Cilk™ Plus Array Notations Semantics Change in update 6

In Intel® C++ Composer XE 2011, a Cilk Plus array section assignment like the following:

```
a[:] = b[:] + c[:];
```

could potentially generate temporary copies of the results, impacting performance.

Starting in Intel® C++ Composer XE 2011 Update 6, if an array section on the right hand side of an assignment (in the example given, b[:] or c[:]) partially overlaps the array section on the left hand side (in the example given, a[:]) in memory, this assignment will be undefined, and it is up to the programmer to assure that there is no partial overlap in memory on assignments in order to get defined behavior.

An exception to this is if the array sections completely overlap, for example:

```
a[:] = a[:] + 3;
```

Since array a overlaps itself completely, this summation will work as expected.

3.1.3 Intel® Cilk™ Plus "scalar" Clause Deprecated

The "scalar" clause used optionally with Cilk Plus elemental functions is being deprecated and will be removed in a future version of C++ Composer XE. Please use the functionally equivalent "uniform" clause instead.

3.1.4 -export and –export-dir deprecated starting in update 4

The two compiler options –export and –export-dir support the C++ template export feature. This feature initially planned for support in the C++0x standard was dropped from this standard. The Intel compiler is deprecating this feature and will remove it in a future release.

3.1.5 Additional Keywords for `-sox` option, default changed in update 3

The `-sox` option, which adds information to the object and executable file about compiler options used and procedure profiling information, has been enhanced to let the user request that the list of inlined functions be included and to let the user exclude information about procedure profiling.

The syntax for `-sox` is now:

```
-[no]sox
-sox=keyword[,keyword]
```

Where *keyword* is one of `inline` or `profile`. If `-sox` is specified with no keywords, only the command line options are included – this is a change from previous releases. To maintain the previous behavior, use `-sox=profile`. Multiple `-sox` options may be specified on the command line – if so, they are interpreted in left-to-right order.

3.1.6 Three intrinsics changed in update 2

Three intrinsics (_rdrand16_step(), _rdrand32_step(), _rdrand64_step()) have been changed in update 2. The documentation has not been updated with these new changes. These intrinsic return a hardware-generated random value and are declared in the "immintrin.h" header file.

These three intrinsics are mapped to a single RDRAND instruction, generate random numbers of 16/32/64 bit wide random integers.

Syntax

1. extern int _rdrand16_step(unsigned short *random_val);
2. extern int _rdrand32_step(unsigned int *random_val);
3. extern int _rdrand64_step(unsigned __int64 *random_val);

Description

> The intrinsics perform one attempt to generate a hardware generated random value using the instruction RDRAND. The generated random value is written to the given memory location and the success status is returned: 1 if the hardware returned a valid random value and 0 otherwise.

Return

A hardware-generated 16/32/64 random value.

Constraints

The _rdrand64_step() intrinsic can be used only on systems with the 64-bit registers support.

3.2 New and Changed Compiler Options

For details on these and all compiler options, see the Compiler Options section of the on-disk documentation.

3.2.1 New and Changed in Composer XE 2011 update 6

- -march=core-avx2
- -xCORE-AVX2
- -axCORE-AVX2
- -march-core-avx-i
- -xCORE-AVX-I
- -axCORE-AVX-I
- -xSSSE3_ATOM
- -masm
- -fasynchronous-unwind-tables
- -opt-mem-layout-trans
- -fopenmp
- -parallel-source-info
- -fgnu89-inline
- -gdwarf-3
- -fno-merge-debug-strings
- -sox
- -fstack-protector-all

3.2.2 New and Changed in Composer XE 2011

- -ansi-alias-check
- -ffriend-injection
- -fzero-initialized-in-bss
- -fimf-absolute-error
- -fimf-accuracy-bits
- -fimf-arch-consistency
- -fimf-max-error
- -fimf-precision
- -fms-dialect
- -fp-trap
- -fp-trap-all
- -fvar-tracking
- -fvar-tracking-assignments

- -opt-args-in-regs
- -prof-value-profiling
- -profile-functions
- -profile-loops
- -regcall
- -simd
- -Wremarks
- -Wsign-compare
- -Wstrict-aliasing

For a list of deprecated compiler options, see the Compiler Options section of the documentation.

3.3 Other Changes

3.3.1 Optimization Reports Disabled by Default

As of version 11.1, the compiler no longer issues, by default, optimization report messages regarding vectorization, automatic parallelization and OpenMP threaded loops. If you wish to see these messages you must request them by specifying `-diag-enable vec`, `-diag-enable par` and/or `-diag-enable openmp`, or by using `-vec-report`, `-par-report` and/or `-openmp-report`.

Also, as of version 11.1, optimization report messages are sent to `stderr` and not `stdout`.

3.3.2 Environment Setup Script Changed

The `compilervars.sh` script is used to establish the compiler environment.

The command takes the form:

`source <install-dir>/bin/compilervars.sh` *`argument`*

Where *`argument`* is either `ia32` or `intel64` as appropriate for the architecture you are building for. Establishing the compiler environment also establishes the environment for the Intel® Debugger, Intel® Performance Libraries and, if present, Intel® Fortran Compiler.

3.3.3 OpenMP* Legacy Libraries Removed

The OpenMP “legacy” libraries have been removed in this release. Only the “compatibility” libraries are provided.

3.4 Known Issues

3.4.1 `__GXX_EXPERIMENTAL_CXX0X__` Macro Not Supported

In the Gnu* version 4.6 or later environments, using the `-std=c++0x` or `-std=gnu++0x` option may lead to a diagnostic of the form:

```
This file requires compiler and library support for the upcoming ISO
C++ standard, C++0x. This support is currently experimental, and must
be enabled with the -std=c++0x or -std=gnu++0x compiler options.
```

The Intel compiler does not currently define the `__GXX_EXPERIMENTAL_CXX0X__` macro in gcc 4.6 mode, since it does not yet support some C++0x features enabled by the macro in the C++ standard library headers. This may lead to incompatibilities with g++ when using the C++ standard library in the `-std=c++0x` or `-std=gnu++0x` modes.

Note that earlier updates of C++ Composer XE 2011 had this issue with gcc versions 4.3, 4.4, and 4.5 as well, but those issues have been resolved and with Update 6, header files provided by these versions of gcc prior to 4.6 should not encounter this problem.

4 Intel® Debugger (IDB)

4.1 Compilation Requirements

Starting with Xcode 2.3, the Dwarf debugging information is stored in the object (.o) files. These object files are accessed by the debugger to obtain information related to the application being debugged and thus must be available for symbolic debugging.

In cases where a program is compiled and linked in one command, such as:

```
icc -g -o hello.exe hello.c
```

the object files are generated by the compiler but deleted before the command completes. The binary file produced by this command will have no debugging information. To make such an application debuggable users have two options.

Users may build the application in two steps, explicitly producing a .o file:

```
icc -c -g -o hello.o hello.c

icc -g -o hello.exe hello.o
```

Alternatively, users may use the compiler switch -save-temps to prevent the compiler from deleting the .o files it generates:

```
icc -g -save-temps -o hello.exe hello.c
```

The debugger does not use the output of the “dsymutil” utility.

4.2 Known Issues

4.2.1 Dwarf vs. Stabs Debug Formats

The debugger only supports debugging of executables whose debug information is in Dwarf2 format, and does not support the Stabs debug format. Use the `-gdwarf-2` flag on the compile

command to have gcc and g++ generate Dwarf output. The Intel compilers (icc and ifort) produce Dwarf2 debug format with the `-g` flag.

4.2.2 Debug Info from Shared Libraries

The debugger does not read debug information from shared libraries. Therefore you cannot set a breakpoint to symbols like _exit which are part of a system library.

4.2.3 Non-local Binary and Source File Access

The debugger cannot access binary files from a network-mounted file system (such as NFS). The error message will look like this:

```
Internal error: cannot create absolute path for: /home/me/hello
You cannot debug "/home/me/hello" because its type is "unknown".
```

The debugger cannot access source files from a network-mounted file system (such as NFS). The error message will look like this:

```
Source file not found or not readable, tried...
./hello.c
/auto/mount/site/foo/usr1/user_me/c_code/hello.c
(Cannot find source file hello.c)
```

The file-path specified will be correct.

The workaround in both cases is to copy the files to a local file system (i.e., one which is not mounted over the network).

4.2.4 Debugging applications that fork

Debugging the child process of an application that calls fork is not yet supported.

4.2.5 Debugging applications that exec

The $catchexecs control variable is not supported.

4.2.6 Snapshots

Snapshots are not yet supported as described in the manual.

4.2.7 Debugging optimized code

Debugging optimized code is not yet fully supported. The debugger may not be able to see some function names, parameters, variables, or the contents of the parameters and variables when code is compiled with optimizations turned on.

4.2.8 Watchpoints

Watchpoints that are created to detect write access don't trigger when a value identical to the original has been written. These restrictions are due to a limitation in the Mac OS* X operating system.

Because the SIGBUS signal rather than the SIGSEGV signal is used by the debugger to implement watchpoints, you cannot create a signal detector which will catch a SIGBUS signal.

4.2.9 Graphical User Interface (GUI)

This version of the debugger does not support the GUI

4.2.10 MPP Debugging Restrictions

MPP debugging is not supported as described in the manual.

4.2.11 Function Breakpoints

Debugger breakpoints set in functions (using the "stop in" command) may not halt user program execution at the first statement. This is due to insufficient information regarding the function prologue in the generated Dwarf debug information. As a work-around, use the "stop at" command to set a breakpoint on the desired statement.

The compiler generates a call to "__dyld_func_lookup" as part of the prologue for some functions. If you set a breakpoint on this function the debugger will stop there, but local variable values may not be valid. The work-around is to set a breakpoint on the first statement inside the function.

4.2.12 Core File Debugging

Debugging core files is not yet supported.

4.2.13 Universal Binary Support

Debugging of universal binaries is supported. The debugger supports debugging the IA-32 Dwarf sections of binaries on IA-32 and either the IA-32 or the Intel® 64 sections on Intel® 64.

4.2.14 Debugger variable $threadlevel

The manual's discussion of the debugger variable "$threadlevel" says "On Mac OS* X, the debugger supports POSIX threads, also known as pthreads." This sentence might be read as implying that other kinds of threads might be supported. This is not true; only POSIX threads are supported on Mac OS* X.

4.2.15 Open File Descriptors Limitation

Because the debugger opens the .o files of a debuggee to read debug information, you should raise the open file limit.

Mac OS* limits the number of open file descriptors to 256. You can increase this limit as follows:

```
ulimit -n 2000
```

Please use this command to increase the number of open descriptors before starting the debugger.

This is a workaround until the debugger can better share a limited number of open file descriptors over many files.

4.2.16 `$cdir, $cwd` Directories

`$cdir` is the compilation directory (if recorded). This is supported in that the directory is set; but `$cdir` is not itself supported as a symbol.

`$cwd` is the current working directory. Neither the semantics nor the symbol are supported.

The difference between `$cwd` and '.' is that `$cwd` tracks the current working directory as it changes during a debug session. '.' is immediately expanded to the current directory at the time an entry to the source path is added.

4.2.17 `info stack` Usage

The GDB mode debugger command "info stack" does not currently support negative frame counts the way GDB does, for the following command:

```
info stack [num]
```

A positive value of num prints the innermost num frames, a zero value prints all frames and a negative one prints the innermost –num frames in reverse order.

4.2.18 `$stepg0` Default Value Changed

The debugger variable `$stepg0` changed default to a value of 0. With the value "0" the debugger will step over code without debug information if you do a "step" command. Set the debugger variable to 1 to be compatible with previous debugger versions as follows:

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

5 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about this version of Intel® Integrated Performance Primitives (Intel® IPP). For detailed information about IPP see the following links:

- **New features**: see the information below and visit the main Intel IPP product page on the Intel web site at: http://www.intel.com/software/products/ipp; and the Intel IPP Release Notes at http://software.intel.com/en-us/articles/intel-ipp-70-library-release-notes/.
- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://www.intel.com/software/products/ipp.

5.1 Intel® IPP Cryptography Libraries are Available as a Separate Download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, please read http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/

5.2 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages at http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/

The samples include source code for audio/video codecs, image processing and media player applications, and for calling functions from C++, C# and Java*. Instructions on how to build the sample are described in a readme file that comes with the installation package for each sample.

6 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about this version of Intel® Math Kernel Library. All the bug fixes can be found here: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

6.1 Changes in This Version

6.1.1 Changes in Initial Release

1) BLAS
   - New functions for computing 2 matrix-vector products at once: [D/S]GEM2VU, [Z/C]GEM2VC
   - New functions for computing mixed precision general matrix-vector products: [DZ/SC]GEMV
   - New function for computing the sum of two scaled vectors: *AXPBY
   - Intel® AVX optimizations in key functions: SMP LINPACK, level 3 BLAS, DDOT, DAXPY
2) LAPACK
   - New C interfaces for LAPACK supporting row-major ordering
   - Integrated Netlib LAPACK 3.2.2 including one new computational routine (*GEQRFP) and two new auxiliary routines (*GEQR2P and *LARFGP) and the earlier LAPACK 3.2.1 update
   - Intel® AVX optimizations in key functions: DGETRF, DPOTRF, DGEQRF
3) PARDISO
   - Improved performance of factor and solve steps in multi-core environments
   - Introduced the ability to solve for sparse right-hand sides and perform partial solves—produces partial solution vector
   - Improved performance of the out-of-core (OOC) factorization step
   - Support for zero-based (C-style) array indexing
   - Zeros on the diagonal of the matrix are no longer required in sparse data structures for symmetric matrices
   - New ILP64 PARDISO interface allows the use of both LP64 and ILP64 versions when linked to the LP64 libraries
   - The memory required for storing files on the disk in OOC mode can now be estimated just after reordering
4) Sparse BLAS
   - Format conversion functions now support all data types (single and double precision for real and complex data) and can return sorted or unsorted arrays
5) FFTs
   - Intel AVX optimizations in all 1D/2D/3D FFTs

- Improved performance of 2D and 3D mixed-radix FFTs for single and double precision data for all systems supporting the SSE4.2 instruction set
- Support for split-complex data represented as two real arrays introduced for 2D/3D FFTs
- Support for 1D complex-to-complex transforms of large prime lengths

6) VML
   - A new function for computing (ax+b)/(cy+d) where a, b, c, and d are scalars, and x and y are real vectors: v[s/d]LinearFrac()
   - Intel AVX optimizations for real functions
   - A new mode for setting denormals to zero, overflow support for complex vectors, and for every VML function a new function with an additional parameter for setting the accuracy mode
7) VSL
   - A set of new Summary Statistics functions was added covering basic statistics, covariance and correlation, pooled, group, partial, and robust covariance/correlation, quantiles and streaming quantiles, outliers detection algorithm, and missing values support
     - Performance optimized algorithms: MI algorithm for support of missing values, TBS algorithm for computation of robust covariance, BACON algorithm for detection of outliers, ZW algorithm for computation of quantiles (streaming data case), and 1PASS algorithm for computation of pooled covariance
   - Improved performance of SFMT19937 Basic Random Number Generator (BRNG)
   - Intel® AVX optimizations: MT19937 and MT2203 BRNGs
8) Added runtime dispatching dynamic libraries allowing link to a single interface library which loads dependent libraries dynamically at runtime depending on runtime CPU detection and/or library function calls
9) The custom dynamic libraries builder now uses the runtime dispatching dynamic libraries on the Linux* and Mac OS* X operating systems
10) A new directory structure has been established to simplify integration of Intel MKL with the Intel® Parallel Studio XE family of products and directories formerly designated as "em64t" are now designated by the "intel64" tag
11) The sparse solver functionality has been fully integrated into the core Intel MKL libraries and the libraries with "solver" in the filename have been removed from the product

6.1.2 Changes in Update 1

1) PARDISO/DSS: Added true F90 overloaded API (see the Intel MKL reference manual for more information)
2) PARDISO: Improved the statistical reporting to be more reader friendly
3) Sparse BLAS: Improved performance of ?BSRMM functions on the latest Intel& processors
4) FFTs: Support for negative strides
5) FFT examples: Added examples for split-complex FFTs in C and Fortran using both the DFTI and FFTW3 interfaces
6) VML: Improved performance of real in-place Add/Sub/Mul/Sqr functions on systems supporting SSE2 and SSE3

7) Poisson Library: Changed the default behavior of the Poisson library functions from sequential to threaded operation
8) Bug fixes: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

6.1.3 Changes in Update 2

1) BLAS: Improved performance of transposition functions on the Intel® Xeon® processor 5600 series
2) BLAS: Added examples for transposition routines
3) FFT: Added Fortran examples showing how to reduce application footprint by linking only functions with the desired precision
4) FFT: Added check for stride consistency on in-place real transforms with CCE storage
5) FFT: Expanded threading to new cases for multi-dimensional transforms
6) VSL: Improved performance of Multivariate Gaussian random number generator for single- and double-precision on 4-core Intel® Xeon® processors 5500 series
7) VML: Improved performance of in-place operation of Add, Mul, and Sub functions on the Intel® Xeon® processor 5500 series
8) Bug fixes: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

6.1.4 Changes in Update 3

1) BLAS: Improved multi-threaded performance of DSYRK, DTRSM, and DGEMM on Intel® Xeon® processor 5400 series running 32-bit Windows*
2) LAPACK: Implemented LAPACK 3.3 from netlib including Cosine-Sine decomposition, improved linear equations solvers for symmetric and Hermitian matrices and auxiliary functions
3) PARDISO: 0-based permutation vectors are now allowed at input
4) PARDISO: Documentation for the pardisoinit() routine
5) PARDISO: Improved performance of serial PARDISO with multiple right-hand sides (RHS)
6) PARDISO: Independent control for parallelism in the solve step for improved performance on small matrices—see description of iparm(25)
7) PARDISO: Reduced backward substitution—allows partial solution computation for a full RHS—see description of iparm(31)
8) FFT: Implemented Real FFT transforms for up to 3 to 7 dimensions
9) FFT: Parallelized multi-dimensional complex transforms using split-complex data represented as two real arrays
10) Cluster FFTs: Extended FORTRAN 90 interface to real-to-complex transforms and included new examples
11) VML: Added new complex Pack/Unpack functions and real Gamma/LGamma functions
12) VML: Improved performance on Intel® Xeon® processor 5600 series and processors supporting Intel® Advanced Vector Extensions (Intel® AVX) for the following: all functions when operating on short vectors (<100), all functions when operating on unaligned input vectors, the sPow2o3 function, and the enhanced performance (EP) version of complex Add and Sub.
13) VSL: Functions for saving/restoring random number generator (RNG) streams to/from memory
14) VSL: Added new UniformBits32 and UniformBits64 functions

15) VSL: Extended the number of unique streams supported by MT2203 BRNG from 1024 to 6024
16) Bug fixes: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

6.1.5 Changes in Update 4

1)BLAS: Improved DTRMM performance on Intel® Xeon® processors 5400 and later
2)BLAS: Improved DTRSM performance on all 64-bit enabled processors, especially processors with Intel® Advanced Vector Extensions (Intel® AVX)
3)LAPACK: Incorporated bug fixes from the LAPACK 3.3.1 release
4)OOC PARDISO: Improved the estimate of the amount of memory needed in out-of-core operation
5)FFT: Improved 1D real FFT scaling through improved threading
6)FFT: Updated C and Fortran FFT examples to use the new single dynamic library linking model
7)VML: Improved performance of the single precision Enhanced Performance version of the real Hypot and complex Abs functions and of the complex Arg, Div, Mul, MulByConj functions for all accuracy modes on Intel® Xeon® processors 5600 and 7500 series, and the Intel® Core™ i7-2600 processor
8)Service functions: Improvements and additions to the Intel MKL service functions (see the online release notes at http://software.intel.com/en-us/articles/intel-mkl-103-release-notes/ for more information)
9)Bug fixes: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

Deprecation Notice

1) The GMP Arithmetic Functions in Intel MKL will be removed in a future version of Intel MKL.
2) The timing function mkl_set_cpu_frequency() is deprecated.

6.1.6 Changes in Update 5

1) BLAS: Improved performance: {S,C,Z}TRSM for processors with Intel® Advanced Vector Extensions (Intel® AVX); {S,D}GEM2VU for processors with Intel AVX as well as the Intel® Core™ i7 processor and the Intel® Xeon® processor 5500 series
2) BLAS: Improved scaling: ?TRMV for large matrices on all architectures; DGEMM for odd numbers of threads on Intel® Xeon® processor 5400 series
3) LAPACK: Included LAPACK 3.3.1 extensions and the respective LAPACKE interfaces
4) LAPACK: Improved the performance of ?SYGST and ?HEGST used in generalized eigenvalue problems
5) LAPACK: Improved the performance of the inverse of an LU factored matrix (?GETRI)
6) PARDISO: Added transpose and conjugate transpose solve capability (ATx=b and AHx=b); facilitates compressed sparse column (CSC) format support
7) PARDISO: Improve out-of-core PARDISO performance when the memory requirements slightly exceed available memory using MKL_PARDISO_OOC_MAX_SWAP_SIZE environment variable and in-core PARDISO
8) Optimization Solvers: Added Inf and NaN checks in the RCI Trust-Region solvers
9) FFTs: Improved the performance of 3D FFTs on small cubes from 2x2x2 to 10x10x10 for all supported precisions and types on all Intel® processors supporting Intel® SSE3 and later

10) FFT examples: Re-designed example programs to cover common use cases for Intel MKL DFTI and FFTW
11) VSL: Improved the performance of the single precision MT19937 and MT2203 basic random number generators on the Intel® Core™ i7-2600 processor on 64-bit operating systems
12) VSL: Improved the performance of the integer version of the SOBOL quasi-random number

6.1.1 Changes in Update 6

1) Sparse BLAS: Added a new option to the mkl_?csrbsr converter function allowing detection and removal of zero elements when converting from the BSR format to the CSR format

6.2 Attributions

As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (http://www.intel.com/software/products/mkl) in both the product documentation and website.

The original versions of the BLAS from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/blas/index.html.

The original versions of LAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/lapack/index.html. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Our FORTRAN 90/95 interfaces to LAPACK are similar to those in the LAPACK95 package at http://www.netlib.org/lapack95/index.html. All interfaces are provided for pure procedures.

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are 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, and R. C. Whaley.

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at http://www.pardiso-project.org.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

7 Intel® Threading Building Blocks

For information on changes to Intel® Threading Building Blocks (Intel® TBB), please read the file `CHANGES` in the Intel® TBB documentation directory.

8 Disclaimer and Legal Information