# Intel® C++ Compiler Professional Edition 11.0 for Linux*

*Installation Guide and Release Notes*
*6 November 2008*

## 1 Introduction

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

### 1.1 Product Contents

*Intel® C++ Compiler Professional Edition 11.0 for Linux** includes the following components:

- Intel® C++ Compilers for building applications that run on IA-32, Intel® 64 and IA-64 architecture systems running the Linux* operating system
- Intel® Debugger
- Intel® Assembler for IA-64 architecture applications
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Threading Building Blocks
- Integration into the Eclipse* development environment
- On-disk documentation

### 1.2 Change History

The following section highlights new features and changes since the initial version 11.0 release. Update numbers may not represent a released update - the indicated and later updates have these changes.

- 11.0.071
  - All documentation for Cluster OpenMP* can now be found at whatif.intel.com
  - The compiler installation on IA-32 architecture systems now verifies the minimum requirement of a processor supporting Intel® Streaming SIMD Extensions 2 (Intel® SSE2)
  - Corrections to reported problems
- 11.0.069
  - Initial release

## 1.3 System Requirements

### 1.3.1 Processor Terminology

Intel® compilers support three platforms: general combinations of processor and operating system type. This section explains the terms that Intel uses to describe the platforms in its documentation, installation procedures and support site.

- **IA-32 Architecture** refers to systems based on 32-bit processors generally compatible with the Intel Pentium® II processor, (for example, Intel® Pentium® 4 processor or Intel® Xeon® processor), or processors from other manufacturers supporting the same instruction set, running a 32-bit operating system ("Linux x86").
- **Intel® 64 Architecture** refers to systems based on IA-32 architecture processors which have 64-bit architectural extensions, (for example, Intel® Core™2 processor family), running a 64-bit operating system ("Linux x86_64"). If the system is running a 32-bit version of the Linux operating system, then IA-32 architecture applies instead. Systems based on AMD* processors running a "Linux x86_64" operating system are also supported by Intel compilers for Intel® 64 architecture applications.
- **IA-64 Architecture** refers to systems based on the Intel® Itanium® processor running a 64-bit operating system.

### 1.3.2 Native and Cross-Platform Development

The term "native" refers to building an application that will run on the same platform that it was built on, for example, building on IA-32 architecture to run on IA-32 architecture. The term "cross-platform" or "cross-compilation" refers to building an application on a platform type different from the one on which it will be run, for example, building on IA-32 architecture to run on IA-64 architecture. Not all combinations of cross-platform development are supported and some combinations may require installation of optional tools and libraries.

The following list describes the supported combinations of compilation host (system on which you build the application) and application target (system on which the application runs).

- IA-32 Architecture Host: Supported target: IA-32
- Intel® 64 Architecture Host: Supported targets: IA-32 and Intel® 64
- IA-64 Architecture Host: Supported target: IA-64

Development for a target different from the host may require optional library components to be installed from your Linux Distribution.

### 1.3.3 Requirements

#### *Requirements to develop IA-32 architecture applications*

- A system based on an IA-32 architecture processor supporting the Intel® Streaming SIMD 2 Extensions (Intel® SSE2) instructions (e.g. Intel Pentium® 4 processor), or an Intel® 64 architecture processor
- 512 MB of RAM (1GB recommended)
- 2GB free disk space for all features

- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Fedora* 9
    - Red Hat Enterprise Linux* 3, 4, 5
    - SUSE LINUX Enterprise Server* 9, 10
    - TurboLinux* 11
    - Ubuntu* 8.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- Linux component compat-libstdc++ providing libstdc++.so.5
- If developing on an Intel® 64 architecture system, Linux component glibc-devel.i386 providing stubs-32.h

### *Requirements to Develop Intel® 64 Architecture Applications*

- A system based on an Intel® 64 architecture processor, or based on an AMD 64-bit processor
- 512 MB of RAM (1GB recommended)
- 2GB free disk space for all features
- 100 MB of hard disk space for the virtual memory paging file. Be sure to use at least the minimum amount of virtual memory recommended for the installed distribution of Linux
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Fedora* 9
    - Red Hat Enterprise Linux* 3, 4, 5
    - SGI ProPack* 5
    - SUSE LINUX Enterprise Server* 9, 10
    - TurboLinux* 11
    - Ubuntu* 8.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- Linux component compat-libstdc++ providing libstdc++.so.5
- Linux component containing 32-bit libraries (may be called ia32-libs)

### *Requirements to Develop IA-64 Architecture Applications*

- A system based on an Intel® Itanium® processor.
- 512 MB of RAM (1 GB recommended).
- 2GB free disk space for all features

- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Red Hat Enterprise Linux* 3, 4, 5
    - SUSE LINUX Enterprise Server* 9, 10
    - TurboLinux* 11
    - Ubuntu* 8.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- Linux component compat-libstdc++ providing libstdc++.so.5

#### *Additional Requirements to use the Graphical User Interface of the Intel® Debugger*

- IA-32 architecture system or Intel® 64 architecture system
- Java* Runtime Environment 5.0 (also called 1.5.0)

#### *Additional Requirements to use Eclipse* Integration*

- IA-32 architecture system or Intel® 64 architecture system
- Eclipse* 3.4.x or 3.3.x
- Eclipse C/C++ Development Tools (CDT) 5.0.0 or 4.0.2
- Java Run-Time Environment 5.0 (1.5.0)

**Notes**

- The Intel compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.
- Compiling very large source files (several thousands of lines) using advanced optimizations such as `-O3`, `-ipo` and `-openmp`, may require substantially larger amounts of RAM.
- The above lists of processor model names are not exhaustive - other processor models correctly supporting the same instruction set as those listed are expected to work. Please refer to Technical Support if you have questions regarding a specific processor model
- Some optimization options have restrictions regarding the processor type on which the application is run. Please see the documentation of these options for more information.

### 1.3.4 Red Hat Enterprise Linux* 3, SUSE LINUX Enterprise Server* 9 Support Deprecated

In a future major release of Intel C++ Compiler, support will be removed for installation and use on Red Hat Enterprise Linux 3 and SUSE LINUX Enterprise Server 9. Intel recommends migrating to a newer version of these operating systems.

## 1.4 Installation

If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

If you received your product on DVD, mount the DVD, change the directory (`cd`) to the top-level directory of the mounted DVD and begin the installation using the command:

```
./install.sh
```

If you received the product as a downloadable file, first unpack it into a writeable directory of your choice using the command:

```
tar –xzvf name-of-downloaded-file
```

Then change the directory (`cd`) to the directory containing the unpacked files and begin the installation using the command:

```
./install.sh
```

Follow the prompts to complete installation.

### 1.4.1 Eclipse* Integration Installation

Please refer to the section below on Eclipse Integration

### 1.4.2 Known Installation Issues

- If you have enabled the Security-Enhanced Linux (SELinux) feature of your Linux distribution, you must change the `SELINUX` mode to `permissive` before installing the Intel C++ Compiler. Please see the documentation for your Linux distribution for details. After installation is complete, you may reset the `SELINUX` mode to its previous value.
- On some versions of Linux, auto-mounted devices do not have the "exec" permission and therefore running the installation script directly from the DVD will result in an error such as:

  ```
  bash: ./install.sh: /bin/bash: bad interpreter: Permission denied
  ```

  If you see this error, remount the DVD with exec permission, for example:

  ```
  mount /media/<dvd_label> -o remount,exec
  ```

  and then try the installation again.
- The Intel C++ Compiler Professional Edition 11.0 for Linux product is fully supported on Ubuntu 8.04 IA-32 and Intel® 64 architecture systems. Due to a restriction in the licensing software, however, it is not possible to use the Trial License feature when evaluating IA-32 components on an Intel® 64 architecture system with Ubuntu 8.04. Earlier versions of Ubuntu, not officially supported by this release of software, may have

similar problems. This affects using a Trial License only. Use of serial numbers, license files, floating licenses or other license manager operations, and off-line activation (with serial numbers) is not affected. If you need to evaluate IA-32 components of the Intel C++ Compiler Professional Edition 11.0 for Linux product on an Intel® 64 architecture Ubuntu system, please visit the Intel Software Evaluation Center to obtain an evaluation serial number.

## 1.5 Installation Folders

The 11.0 product installs into a different arrangement of folders than in previous versions. The new arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- `<install-dir>/Compiler/11.0/`*`xxx/`*
    - `bin`
        - `ia32`
        - `intel64`
        - `ia64`
    - `include`
        - `ia32`
        - `intel64`
        - `ia64`
    - `perf_headers`
    - `substitute_headers`
    - `lib`
        - `ia32`
        - `intel64`
        - `ia64`
    - `eclipse_support`
    - `idb`
        - `eclipse_support`
        - `gui`
        - `ia32`
        - `ia64`
        - `intel64`
        - `lib`
        - `third_party`
    - `ipp`
        - `em64t`
        - `ia32`
        - `ia64`
    - `mkl`
        - `benchmarks`
        - `examples`
        - `include`

        - `interfaces`
        - `lib`
        - `tests`
        - `tools`
    - `tbb`
        - `bin`
        - `em64t`
        - `examples`
        - `ia32`
        - `include`
        - `itanium`
        - `lib`
    - `Documentation`
        - `cluster_omp`
        - `compiler_c`
        - `en_US`
        - `idb`
        - `ipp`
        - `ja_JP`
        - `mkl`
        - `tbb`
    - `man`
    - `Samples`
    - `cluster_omp`

Where `<install-dir>` is the installation directory (default for system-wide installation is `/opt/intel`) and *`xxx`* is the three-digit update number and the folders under `bin`, `include` and `lib` are used as follows:

- `ia32`: Files used to build applications that run on IA-32
- `intel64`: Files used to build applications that run on Intel® 64
- `ia64`: Files used to build applications that run on IA-64

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

## 1.6 Removal/Uninstall

Removing (uninstalling) the product should be done by the same user who installed it (root or a non-root user). It is not possible to remove the compiler while leaving any of the performance library or Eclipse* integration components installed.

1. Open a terminal window and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command: `<install-dir>/bin/ia32/uninstall_cproc.sh` (substitute `intel64` or `ia64` for `ia32` as desired)

3. Follow the prompts
4. Repeat steps 2 and 3 to remove additional platforms or versions

If you have the same-numbered version of Intel® Fortran Compiler installed, it may also be removed. If you have added the Intel C++ Eclipse integration to an instance of Eclipse in your environment, you will need to update your Eclipse configuration by removing the Intel integration extension site from your Eclipse configuration.

### 1.7 Documentation

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

### 1.8 Technical Support

Register your license 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/clin

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

## 2 Intel® C++ Compiler

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

### 2.1 Compatibility

In version 11, the IA-32 architecture default for code generation has changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. See below for more information.

### 2.2 New and Changed Features

Please refer to the compiler documentation for details

- Additional features from C++ 0x
- C++ lambda functions
- Decimal floating point
- valarray implementation using IPP option
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- Support for OpenMP* 3.0

- The default mode of the C++ compiler now more closely matches the default mode of gcc. Some C99 features, such as mixed declarations and code, may no longer be turned on by default, but can be enabled using -std=c99

## 2.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- -axSSE2
- -axSSE3
- -axSSSE3
- -axSSE4.1
- -axSSE4.2
- -diag-error-limit
- -diag-once
- -falign-stack
- -fast-transcendentals
- -ffreestanding
- -finline
- -fma
- -fnon-call-exceptions
- -fp-relaxed
- -fpie
- -fstack-protector
- -help-pragma
- -m32
- -m64
- -mia32
- -minstruction
- -mssse3
- -msse4.1
- -multiple-processes
- -openmp-link
- -openmp-task
- -openmp-threadprivate
- -opt-block-factor
- -opt-calloc
- -opt-jump-tables
- -opt-loadpair
- -opt-mod-versioning
- -opt-prefetch-initial-values
- -opt-prefetch-issue-excl-hint
- -opt-prefetch-next-iteration
- -opt-subscript-in-range

- -pie
- -prof-data-order
- -prof-func-order
- -prof-hotness-threshold
- -prof-src-root
- -prof-src-root-cwd
- -tcollect-filter
- -vec
- -Werror-all
- -Wformat-security
- -xHost
- -xL
- -xSSE2
- -xSSE3
- -xSSSE3
- -xSSE4.1
- -xSSE4.2

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

### 2.3.1 `-xHost` Option

The `–xHost` option, new in version 11.0, automatically selects the instruction set usage based on the type of processor present in the system used to compile the source. The behavior is as follows:

| **Processor in compiling system** | **Implied option** |
|---|---|
| Intel processor supporting Intel® Streaming SIMD Extensions 2 (Intel® SSE2) or higher instructions | `-xSSE4.2`, `-xSSE4.1`, `-xSSSE3`, `-xSSE3` or `-xSSE2` as appropriate |
| Older Intel processor | `-mia32` |
| Non-Intel processor | `-mSSE3`, `-mSSE2` or `–mia32` as appropriate, based on the capabilities claimed by the processor |

When using the instruction set options, make sure that the executing system supports the specified instruction set. `–xHost` is best used when the same system will be used to compile and run the application.

## 2.4 Other Changes

### 2.4.1 Establishing the Compiler Environment

The `iccvars.sh` (`iccvars.csh`) script, used to set up the command-line build environment, has changed. In previous versions, you chose the target platform by selecting either the `cc` or `cce` directory root.  In version 11.0, there is one version of these scripts and they now take an argument to select the target platform.

The command takes the form:

```
source <install-dir>/Compiler/11.0/xxx/bin/iccvars.sh argument
```

Where `<install-dir>` is the installation directory (default for system-wide installation is `/opt/intel`) and *`xxx`* is the update number and *`argument`* is one of `ia32`, `intel64`, `ia64` as described above under Installation Folders. Establishing the compiler environment also establishes the Intel® Debugger (idb) environment.

### 2.4.2 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)

When compiling for the IA-32 architecture, `-msse2` (formerly `-xW`) is now the default.  Programs built with `–msse2` in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and certain AMD* processors. No run-time check is made to ensure compatibility – if the program is run on an unsupported processor, an invalid instruction fault may occur.  Note that this may change floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify `–mia32`

### 2.4.3 OpenMP* Libraries Default to “compat”

In version 10.1, a new set of OpenMP libraries was added that allowed applications to use OpenMP* code from both Intel and Microsoft compilers.  These “compatibility” libraries can provide higher performance than the older “legacy” libraries.  In version 11.0, the compatibility libraries are used by default for OpenMP applications, equivalent to `-openmp-lib compat`.  If you wish to use the older libraries, specify `-openmp-lib legacy`

The “legacy” libraries will be removed in a future release of the Intel compilers.

### 2.4.4 `mathf.h` No Longer Provided

The header file `mathf.h`, used to define single-precision math library functions, has been removed from the product.  If you were using this header file, please use `mathimf.h` instead.

### 2.4.5 Sampling-based Profile Guided Optimization Feature Removed

The hardware sampling-based Profile-Guided Optimization feature is no longer provided. The `-prof-gen-sampling` and `-ssp` compiler options and the `profrun` and `pronto_tool` executables have been removed. Instrumented Profile-Guided Optimization is still supported.

## 2.5 Using Static Verifier in the Eclipse* IDE

When Static Verifier support is enabled within the IDE, the customary final build target (e.g. an executable image) is not created. As such, we recommend that a separate "Static Verification" configuration be created, by cloning the existing Debug (development) configuration, for use when static verification is desired.

- Open the property pages for the project and select "C/C++ Build".
- Click the "Manage" button
- In the "Manage" dialog, click the "New" button to open the "Create configuration" dialog.
- Supply a name for the new configuration in the "Name" box.
- Supply a "Description" for the configuration if you want (optional).
- You can choose to "Copy settings from" a "Default configuration" or an "Existing configuration" by clicking the appropriate radio button and then selecting a configuration from the corresponding drop down menu.
- Click "O.K." to close the "Create configuration" dialog.
- Click "O.K." to close the "Manage" dialog (with your new configuration name selected).
- The property pages will now be displaying the settings for your new configuration and it is now the active build configuration.
- Navigate to the Intel compiler's Compilation Diagnostics properties. Use the "Level of Static Analysis" and "Analyze Included Files" properties to control Static Verification.

## 2.6 Known Issues

### 2.6.1 TR1 System Headers

If you are using the TR1 (C++ Library Technical Report 1) system headers on a system with g++ version 4.3 or later installed, the Intel C/C++ compiler will give errors when it tries to compile the <type_traits> header file. This is because the Intel C/C++ compiler does not yet support the C++0x feature called variadic templates. You will see these types of compilation errors:

```
../include/c++/4.3.0/tr1_impl/type_traits(170): error: expected an
identifier
   template<typename _Res, typename... _ArgTypes>
                                    ^

include/c++/4.3.0/tr1_impl/type_traits(171): error: expected a ")"
    struct __is_function_helper<_Res(_ArgTypes...)>
```

There is no workaround, other than not using these headers or using an older version of the g++ compiler.

### 2.6.2 The behavior default behavior for `KMP_AFFINITY` has changed

The thread affinity type of the `KMP_AFFINITY` environment variable defaults to none (`KMP_AFFINITY=none`). The behavior for `KMP_AFFINITY=none` was changed in 10.1.015 or later, and in all 11.0 compilers, such that the initialization thread creates a "full mask" of all the threads on the machine, and every thread binds to this mask at startup time.  It was subsequently found that this change may interfere with other platform affinity mechanisms, for example, `dplace()` on SGI Altix machines.  To resolve this issue, a new affinity type `disabled` was introduced in compiler 10.1.018, and in all 11.0 compilers (`KMP_AFFINITY=disabled`).  Setting `KMP_AFFINITY=disabled` will prevent the OpenMP runtime library from making any affinity-related system calls.

# 3 Intel® Debugger (IDB)

The following notes refer to a new Graphical User Interface (GUI) available for the Intel® Debugger (IDB) when running on IA-32 and Intel® 64 architecture systems. In this version, the `idb` command invokes the GUI – to get the command-line interface, use `idbc`.

On IA-64 architecture systems, the GUI is not available and the `idb` command invokes the command-line interface.

## 3.1 Setting up the Java Runtime Environment

The Intel® IDB Debugger graphical environment is a Java application and requires a Java Runtime Environment (JRE) to execute. The debugger will run with a version 5.0 (also called 1.5) JRE.

Install the JRE according to the JRE provider's instructions.

Finally you need to export the path to the JRE as follows:

```
    export PATH=<path_to_JRE_bin_dir>:$PATH
```

## 3.2 Starting the Debugger

To start the debugger, first make sure that the compiler environment has been established as described at Establishing the Compiler Environment. Then use the command:

```
idb
```

or

```
idbc
```

as desired.

Once the GUI is started and you see the console window, you're ready to start the debugging session.

Note: Make sure that the executable you want to debug is built with debug info and is an executable file. Change permissions if required, e.g. `chmod +x <application_bin_file>`

## 3.3 Additional Documentation

Online help titled *Intel® Compilers / Intel® Debugger Online Help* is accessible from the debugger graphical user interface as `Help > Help Contents`.

Context-sensitive help is also available in several debugger dialogs where a `Help` button is displayed.

## 3.4 Debugger Features

### 3.4.1 Main Features of IDB

The debugger supports all features of the command line version of the Intel® IDB Debugger. Debugger functions can be called from within the debugger GUI or the GUI-command line. Please refer to the Known Limitations when using the graphical environment.

### 3.4.2 New and Changed Features

- Debugger GUI for IA-32 and Intel® 64 architectures
- Parallel Execution Debug Support
- Session Concept
- Bitfield editor
- SIMD (MMX) register window
- OpenMP information windows
- Internationalization support
- OpenMP configuration support
- Cluster OpenMP Support

## 3.5 Known Problems

### 3.5.1 Signals Dialog not working

The Signals dialog accessible via the GUI dialog Debug / Signal Handling or the shortcut Ctrl+S is not working correctly. Please refer to the Intel® Debugger (IDB) Manual for use of the signals command line commands instead.

### 3.5.2 Debugging Shared Libraries

Debugging applications that load shared libraries on runtime may cause the error:

Error: could not start debugee

Could not start process for <executable>

No image loaded … Recovering …

Even exporting the environment variable LD_LIBRARY_PATH to the directory where the shared library is located may not help. The error message is misleading as well. The debuggee is started, but the debugger cannot find the associated shared library/libraries.

### 3.5.3 `list` command
In GDB mode, unquoted filenames do not work. The workaround is to use quoted filenames, e.g. `list "test.c":10`

### 3.5.4 Resizing GUI
If the debugger GUI window is reduced in size, some windows may fully disappear. Enlarge the window and the hidden windows will appear again.

### 3.5.5 Kill Process
The 'Kill Focused Process' command from the Debug menu does not work when the debugger is running. Stop the debugger first and then kill the process.

### 3.5.6 Serialize Parallel Regions
The 'Serialize Parallel Region' toolbar icon in the GUI does not update correctly. It actually is a toggle button that indicates with a frame around the icon when the Serialize Parallel Regions feature is activated; if there is no frame it is deactivated. The button does not get displayed correctly. After a step/run-stop etc the button is always displayed in disabled mode, even if the feature is activated. It is a display problem only that vanishes if you place the mouse over the icon (then it shows the state correctly until the next step).

The Serialize Parallel Regions option works correctly when displayed from the 'Parallel' menu; the checkmark is correctly set if activated and not set if deactivated.

### 3.5.7 Decimal Floating Point Not Supported
The Intel® Debugger does not support decimal floating point data types, supported in some C++ compilers.  The debugger will display such values as if they were arrays of characters.

### 3.5.8 OpenMP Locks: "No information available"
The Locks, Barriers, and Taskwaits windows always show "No information available" because the OpenMP runtime library is not able to provide the information on these objects in this release. You can get the information for a lock through the command line in the Console window by using this command:

```
    idb info lock <lock_id>
```

where <`lock_id`> is the name of the lock in the program.

### 3.5.9 Online Help Error "Unable to open web browser"
When accessing the on-disk help from IDB, the error "Unable to open web browser on {0}" may occur with certain Linux distributions.  This will happen if the Mozilla* web browser is not found.  A workaround is to create a symbolic link to an installed browser such as Firefox*.  For example:

```
sudo ln -s /usr/bin/firefox /usr/bin/mozilla
```

or, if you do not have sudo root rights:

```
ln -s /usr/bin/firefox <user_dir>/mozilla
```

and add `<user_dir>/Mozilla` to `$PATH`.

### 3.5.10 Online Help on Fedora* 9 Systems

On Fedora* 9 systems, online help is not available from:

- The GUI menu `Help` > `Help Contents`
- The context-sensitive `Help` button in debugger dialogs

If this issue affects you, please start the Debugger help from:

```
  <install-dir>/Compiler/11.0/xxx/idb/gui/common/webhelp/index.htm
```

# 4 Eclipse Integration

The Intel C++ Compiler for the IA-32 and Intel® 64 architectures installs an Eclipse feature and associated plugins (the Intel C++ Eclipse Product Extension) which provide support for the Intel C++ compiler when added as an Eclipse product extension site to an existing instance of the Eclipse* Integrated Development Environment (IDE). With this feature, you will be able to use the Intel C++ compiler from within the Eclipse integrated development environment to develop your applications.

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt5.0/eclipse
```

supports and requires Eclipse Platform version 3.4.x, Eclipse C/C++ Development Tools (CDT) version 5.0.0 or later and a functional Java Runtime Environment (JRE) (minimum version 5.0 (also called 1.5), recommended version 5.0).

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt4.0/eclipse
```

supports and requires Eclipse Platform version 3.3.x, Eclipse C/C++ Development Tools (CDT) version 4.0.2 or later and a functional Java Runtime Environment (JRE) (minimum version 1.4.2, recommended version 5.0 (1.5)).

Note that the Eclipse Platform versions 3.3 and 3.4 are not currently available for the IA-64 architecture. The compiler kit includes an Eclipse integration for that architecture should the platform be released at a later date.

If you already have the proper versions of Eclipse, CDT and a functional JRE installed and configured in your environment, then you can add the Intel C++ Eclipse Product Extension to your Eclipse Platform, as described in the section, below, entitled How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform. Otherwise, you will first need to obtain and install Eclipse, CDT and a JRE, as described in the section, below, entitled How to Obtain and Install Eclipse, CDT and a JRE and then install the Intel C++ Eclipse Product Extension.

## 4.1 How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform

To add the Intel C++ product extension to your existing Eclipse configuration, follow these steps, from within Eclipse.

### 4.1.1 Eclipse 3.4.0 and CDT 5.0.0 “Ganymede”

Open the "Software Updates and Add-ons" page by selecting:

```
Help > Software Updates...
```

Select the "Available Software" tab.

Select "Add Site..." and then "Local...". A directory browser will open. Browse to select the `eclipse` directory in your Intel C++ compiler installation. For example, if you installed the compiler as root to the default directory, you would browse to `/opt/intel/Compiler/11.0/uuu/eclipse_support/cdt5.0/eclipse`.

Select "OK" to close the directory browser. Then select "OK" to close the “Add Site” dialog. Select the two boxes for the Intel C++ integration: there will be one box for “Intel® C++ Compiler Documentation” and a second box for “Intel® C++ Compiler Professional 11.0 for Linux*”. Then click the “Install” button. An “Install” dialog will open which gives you a chance to review and confirm you want to install the checked items. Click “Next”. You will now be asked to accept the license agreement. Accept the license agreement and click “Finish”. The installation of the Intel support will proceed.

When asked to restart Eclipse, select “Yes”. When Eclipse restarts, you will be able to create and work with CDT projects that use the Intel C++ compiler. See the Intel C++ Compiler documentation for more information. You can find the Intel C++ documentation under `Help > Help Contents > Intel C++ Compiler Users Guide`.

If you also installed the Intel® Debugger (idb) along with the idb Eclipse product extension, and would like to use idb within Eclipse, you should add the idb product extension site to your Eclipse configuration in the same way. For example, if you installed the kit as root to the default directory, you would find the idb Eclipse product extension site at `/opt/intel/Compiler/11.0/uuu/idb/eclipse_support/cdt5.0/eclipse`.

### 4.1.2 Eclipse 3.3.2 and CDT 4.0.3 “Europa”

Open the "Product Configuration" page by selecting:

```
 Help > Software Updates > Manage Configuration
```

Under `Available Tasks`, select `Add An Extension Location`. A directory browser will open. Browse to select the `eclipse` directory in your Intel C++ compiler installation. For example, if you installed the compiler as root to the default directory, and you wish to add the Intel C++ compiler support to the latest Eclipse platform release, Eclipse 3.3.2, you would browse to `/opt/intel/Compiler/11.0/uuu/eclipse_support/cdt4.0/eclipse`.

When asked to restart Eclipse, select `Yes`. When Eclipse restarts, you will be able to create and work with CDT projects that use the Intel C++ compiler. See the Intel C++ Compiler documentation for more information.

If you also installed the Intel® debugger (idb) along with the idb Eclipse product extension, and would like to use idb within Eclipse, you should add the idb product extension site to your Eclipse configuration in the same way. For example, if you installed the kit as root to the default directory, you would find the idb Eclipse product extension site at `/opt/intel/Compiler/11.0/uuu/idb/eclipse_support/cdt4.0/eclipse`.

## 4.2 How to Obtain and Install Eclipse, CDT and a JRE

Eclipse is a Java application and therefore requires a Java Runtime Environment (JRE) to execute. The 3.3.2 version of the Eclipse platform will run with a minimum version 4.2 JRE (also called 1.4.2). The 3.4.0 version of the Eclipse platform will run with a minimum version 5.0 JRE. Intel recommends using a 5.0 (1.5) JRE. The choice of a JRE is dependent on your operating environment (machine architecture, operating system, etc.) and there are many JRE's available to choose from.

### 4.2.1 For IA-32 Architecture Users

#### *4.2.1.1 Eclipse 3.4.0 and CDT 5.0.0*

Please visit the Eclipse Foundation website. Click on the `Download Eclipse` button. This will take you to the Eclipse Downloads page. Under "Eclipse Ganymede Packages," find "Eclipse IDE For C/C++ Developers" and select the "Linux 32bit" link for this download package. This will take you to the Eclipse Downloads - mirror selection page. You will be downloading a package named `eclipse-cpp-ganymede-linux-gtk.tar.gz`. Click on a mirror site close to you and save the `tar` file to your machine.

#### *4.2.1.2 Eclipse 3.3.2 and CDT 4.0.3*

Please visit http://www.eclipse.org/downloads/packages/release/europa/winter. Under "Eclipse Europa Packages," find "Eclipse IDE for C/C++ Developers" and select the "Linux 32bit" link for this download package. This will take you to the Eclipse Downloads - mirror selection page. You will be downloading a package currently named `eclipse-cpp-europa-winter-linux-gtk.tar.gz`. The name may change over time as further maintenance releases are posted on eclipse.org. Click on a mirror site close to you and save the `tar` file to your machine.

### 4.2.2 For Intel® 64 Architecture Users

#### *4.2.2.1 Eclipse 3.4.0 and CDT 5.0.0*

Please visit the Eclipse Foundation website. Click on the `Download Eclipse` button. This will take you to the Eclipse Downloads page. Under "Eclipse Ganymede Packages," find "Eclipse IDE For C/C++ Developers" and select the "Linux 64bit" link for this download package. This will take you to the Eclipse Downloads - mirror selection page. You will be downloading a package named `eclipse-cpp-ganymede-linux-gtk-x86_64.tar.gz`. Click on a mirror site close to you and save the `tar` file to your machine.

#### 4.2.2.2 Eclipse 3.3.2 and CDT 4.0.3

The Europa bundled packages do not include the x86_64 components. You will need to obtain the Eclipse platform runtime binary and CDT feature as two separate downloads. The Eclipse platform version 3.3.2 runtime binary tar file is named `eclipse-platform-3.3.2-linux-gtk-x86_64.tar.gz` and can be downloaded from here:

http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.3.2-200802211800/eclipse-platform-3.3.2-linux-gtk-x86_64.tar.gz

To obtain the CDT 4.0.3 feature, please visit http://download.eclipse.org/tools/cdt/releases/europa/ . Click on the

`4.0.3 (Feb 26, 2008)` link. The date may change over time. This will take you to the `Eclipse downloads - mirror selection` page. Click on a mirror site close to you and save the zip file to your machine. The zip file is named `cdt-master-4.0.3.zip`.

### 4.2.3 Installing JRE, Eclipse and CDT

Once you have downloaded the appropriate files for Eclipse, CDT, and a JRE, you can install them as follows:

1. Install your chosen JRE according to the JRE provider's instructions.
2. Create a directory where you would like to install Eclipse and `cd` to this directory. This directory will be referred to as `<eclipse-install-dir>`
3. For IA-32 architecture users and Intel® 64 architecture users using Ganymede:
    a. Copy the Eclipse Ganymede/Europa tar file to the |<eclipse-install-dir>| directory.
    b. Expand the tar file. For example:

    ```
    tar zxvf eclipse-cpp-ganymede-linux-gtk.tar.gz
    ```

    c. Start Eclipse.
4. For Intel® 64 architecture users using Europa:
    a. Copy the Eclipse Europa platform binary runtime tar file to the `<eclipse-install-dir>` directory.
    b. Copy the Eclipse CDT 4.0.3 master zip archive file to the `<eclipse-install-dir>` directory.

Expand the Eclipse Platform Runtime Binary tar file. For example:

```
tar zxvf eclipse-platform-3.3.2-linux-gtk-x86_64.tar.gz
```

    c. Start Eclipse and select `Help->Software Updates->Find and Install`.
    d. Select `Search for new features to install` and click `Next`. Click on `New Archived Site`, select the `cdt-master-4.0.3.zip` file you downloaded, and click `OK`.

e. Click `OK` again. The `cdt-master-4.0.3.zip` will be added to the "Sites to include in search:".
f. Click `Finish`. This will lead to `Updates` window. Expand the entry for `cdt-master-4.0.3.zip` archive. Then expand the entry for the CDT `4.0.3.200802251018` update site and select the `Eclipse C/C++ Development Tools` feature from the list of available features.
g. Click `Next` and accept the license terms. Click `Finish` and then `Install`. Restart Eclipse as prompted.

You are now ready to add the Intel C++ product extension to your Eclipse configuration as described in the section, How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform. If you need help with launching Eclipse for the first time, please read the next section.

## 4.3 Launching Eclipse for Development with the Intel C++ Compiler

If you have not already set your `LANG` environment variable, you will need to do so. For example,

```
setenv LANG en_US
```

Setup Intel C++ compiler related environment variables by executing the `iccvars.csh` (or `.sh`) script prior to starting Eclipse:

`source <install-dir>/bin/iccvars.csh arch_arg` (where "arch_arg" is one of "ia32" or "intel64").

Since Eclipse requires a JRE to execute, you must ensure that an appropriate JRE is available to Eclipse prior to its invocation. You can set the `PATH` environment variable to the full path of the folder of the `java` file from the JRE installed on your system or reference the full path of the java executable from the JRE installed on your system in the `-vm` parameter of the Eclipse command, e.g.:

```
eclipse -vm /JRE folder/bin/java
```

Invoke the Eclipse executable directly from the directory where it has been installed. For example:

```
<eclipse-install-dir>/eclipse/eclipse
```

## 4.4 Installing on Fedora* Systems

If the Intel C++ Compiler for Linux is installed on an IA-32 or Intel® 64 architecture Fedora* system as a "local" installation, i.e. not installed as root, the installation may fail to properly execute the Eclipse graphical user interfaces to the compiler or debugger. The failure mechanism will typically be displayed as a `JVM Terminated` error. The error condition can also occur if the software is installed from the root account at the system level, but executed by less privileged user accounts.

The cause for this failure is that a more granular level of security has been implemented on Fedora, but this new security capability can adversely affect access to system resources, such as dynamic libraries. This new *SELinux* security capability may require adjustment by your system administrator in order for the compiler installation to work for regular users.

### 4.5 Selecting Compiler Versions

For Eclipse projects you can select among the installed versions of the Intel C++ Compiler. On IA-32 architecture systems, the supported Intel compiler versions are 9.1, 10.0, 10.1 and 11.0. On Intel® 64 architecture systems, only compiler version 11.0 is supported.

## 5 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about Intel® Integrated Performance Primitives (Intel® IPP).

### 5.1 Change History

- New function implementation in Image Processing domain ippiCopy* and ippiTranspose* functions
- Other new function implementations in speech coding and signal processing domains. Check "NewFunctionsList.txt" in the documentation directory for more details
- New unified image codec (UIC) frameworks implementation to standardize the interfaces as plug-and-play of various image codecs
- Intel® Atom™ Processor support
- High-level Data Compression library Support lzo and new continued performance improvement for zlib, gzip, bzip2
- A new sample for DMIP Deferred Mode of Image Processing over Intel IPP binary and API
- Intel® QuickAssist functional API for Cryptography
- New Domain - Data Integrity Functions based on operations over finite fields for error-correcting coding
- Generated domain/functionality (Spiral)
- Video Enhancement Denoising / Deinterlasing / Demosaicing
- Image Search descriptors (MPEG7), Color layout, Edge Histogram
- Microsoft RT Audio Support (enchancement)
- New Speech Coding Standard G729.1 Codec Support
- Super Resolution Technology, Optical Flow
- New Video AVS Codec Support for Decoding
- New Image Processing functions for 3D Support, Geom WarpAffine
- New Cryptography function support for Reed-Solomon Algorithm
- Threaded Static Libraries
- ALS Decoder Profile support in AAC Decoding

# 6 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about Intel® Math Kernel Library (Intel® MKL).

## 6.1 New and Changed Features

- Performance Improvements in the BLAS:
    - 32-bit improvements
        - 40-50% improvement for (Z,C)GEMM on Quad-Core Intel® Xeon® processor 5300 series
        - 10% improvement for all GEMM code on Quad-Core Intel® Xeon® processor 5400 series
    - 64-bit improvements
        - 2.5-3% improvement for DGEMM on 1 thread on Quad-Core Intel® Xeon® processor 5400 series
        - 50% improvement for SGEMM on the Intel® Core™ i7 processor family
        - 3% improvement for CGEMM on 1 thread on the f Intel® Core™ i7 processor family
        - 2-3% improvement for ZGEMM on 1 thread on the  Intel® Core™ i7 processor family
        - 30% improvement for right-side cases of DTRSM on the Intel® Core™ i7 processor family
- Improvements to the direct sparse solver (DSS/PARDISO):
    - The performance of out-of-core PARDISO was improved by 35% on average.
    - Support of separate backward/forward substitution for DSS/PARDISO has been added.
    - A new parameter for turning off iterative refinement for DSS interface has been introduced.
    - A new parameter for checking sparse matrix structure has been introduced for PARDISO interface.
- The capability to track the progress of a lengthy computation and/or interrupt the computation has been added via a callback function mechanism. A function called mkl_progress can be defined in a user application, which will be called regularly from a subset of the MKL LAPACK routines. See the LAPACK Auxiliary and Utility Routines chapter in the reference manual for more information. Refer to the specific function descriptions to see which LAPACK functions support the feature.
- Transposition functions have been added to Intel MKL. See the reference manual for further detail.
- The C++ std::complex type can now be used instead of MKL-specific complex types.
- An implementation of the Boost uBLAS matrix-matrix multiplication routine is now provided which will make use of the highly optimized version of DGEMM in the Intel MKL BLAS. See the User guide for more information.
- Improvements to the sparse BLAS:
    - Support for all data types (single precision, complex and double complex) has been added.

    - Routines for computing the sum and product of two sparse matrices stored, both stored in the compressed sparse row format have been added.
- The Vector Math Library functions, CdfNorm, CdfNormInv, and ErfcInv, have been optimized to achieve much improved performance.
- Performance improvement on the Intel® Core™ i7 processor family:
    - 3-17% improvement for the following VML functions: Asin, Asinh, Acos, Acosh, Atan, Atan2, Atanh, Cbrt, CIS, Cos, Cosh, Conj, Div, ErfInv, Exp, Hypot, Inv, InvCbrt, InvSqrt, Ln, Log10, MulByConj, Sin, SinCos, Sinh, Sqrt, Tanh.
    - 7-67% improvement for uniform random number generation.
    - 3-10% improvement for VSL distribution generators based on Wichmann-Hill, Sobol, and Niederreiter BRNGs (64-bit only).
- The configuration file functionality has been removed. See the user guide for alternative means to configure the behavior of Intel MKL.
- When functions in Intel MKL are called from an MPI program they will be run on 1 thread by default (i.e., in the absence of explicit controls).
- The following VML functions: CdfNorm, CdfNormInv, and ErfcInv.
- The DftiCopyDescriptor function.
- The LP64 interface of DSS/PARDISO now uses 64-bit addressing for internal arrays on 64-bit operating systems. This allows the direct solver to solve larger systems.
- The default OpenMP runtime library for Intel MKL has been changed from libguide to libiomp. See the User Guide in the doc directory for more information.
- The optimized code paths for the Intel® Pentium® III processor have been removed from Intel MKL along with the associated processor specific dynamic link libraries. We continue to support the use of Intel MKL on this processor, but the default code path will be used and as a result performance may be reduced.
- The interval linear solver functions have been removed from MKL.
- Support for Intel MPI 1.x has ended.
- Documentation updates:
    - Eclipse IDE Infopop support for VML functions and VSL service functions. The infopop support means brief info on a function in a pop-up window appearing when the cursor is placed to the function/routine name in the Eclipse Editor panel. This Eclipse feature is implemented in the CDT 5.0 version.
    - The FFTW Wrappers for MKL Notes have been removed from the product package after their content was integrated into the Intel MKL Reference Manual (Appendix G).
    - New functions have been documented in the reference manual, and support for Boost uBLAS matrix-matrix multiplication has been described in the User Guide.
    - The parallel BLAS (PBLAS) which support ScaLAPACK are now documented in the Intel MKL reference manual.
    - Added FORTRAN 77 support info to the description of VML and VSL functions in the Intel MKL reference manual.

## 6.2 Known Limitations

### 6.2.1 Limitations to the sparse solver and optimization solvers:

- Sparse and optimization solver libraries functions are only provided in static form

### 6.2.2 Limitations to the FFT functions:

- Mode DFTI_TRANSPOSE is implemented only for the default case

- Mode DFTI_REAL_STORAGE can have the default value only and cannot be set by the DftiSetValue function (i.e. DFTI_REAL_STORAGE = DFTI_REAL_REAL)

- The ILP64 version of Intel® MKL does not currently support FFTs with any one dimension larger than 2^31-1. Any 1D FFT larger than 2^31-1 or any multi-dimensional FFT with any dimension greater than 2^31-1 will return the "DFTI_1D_LENGTH_EXCEEDS_INT32" error message. Note that this does not exclude the possibility of performing multi-dimensional FFTs with more than 2^31-1 elements; as long as any one dimension length does not exceed 2^31-1

- Some limitations exist on arrays sizes for Cluster FFT functions. See mklman.pdf for a detailed description

- When a dynamically linked application uses Cluster FFT functionality, it is required to put the static Intel® MKL interface libraries on the link line as well. For example: -Wl,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -Wl,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread

### 6.2.3 Limitations to the LAPACK functions:

- The ILAENV function, which is called from the LAPACK routines to choose problem-dependent parameters for the local environment, cannot be replaced by a user's version

- second() and dsecnd() functions may not provide correct answers in the case where the CPU frequency is not constant.

### 6.2.4 Limitations to the Vector Math Library (VML) and Vector Statistical Library (VSL) functions:

- Usage of mkl_vml.fi may produce warning about TYPE ERROR_STRUCTURE length

### 6.2.5 Limitations to the ScaLAPACK functions:

- The user can not substitute PJLAENV for their own version. This function is called by ScaLAPACK routines to choose problem-dependent parameters for the local environment.

- ScaLAPACK libraries are available only in static form

### 6.2.6 Limitations to the ILP64 version of Intel® MKL:

- The ILP64 version of Intel® MKL does not contain the complete functionality of the library. For a full listing of what is in the ILP64 version refer to the user's guide in the doc directory.

- g77 cannot be used with the ILP64 libraries.

### 6.2.7 Limitations to the Fortran 95 interface to LAPACK:

- If you are compiling the Fortran 95 interface to LAPACK with the GNU gfortran compiler, you must manually remove the "pure" attribute from all subroutines containing a procedure argument: ?GEES, ?GEESX, ?GGES, ?GGESX (where ? can be S, D, C, or Z).

### 6.2.8 Limitations to the g77 compiler support:

- Some Intel® MKL functions contain underscore in their names (i.e. mkl_dcsrsymv, mkl_cspblas_dcsrsymv) and these functions don't support the g77 default naming convention. -fno-second-underscore compilation flag can be used as workaround for this limitation. E.g.: g77 -fno-second-underscore test.f

### 6.2.9 Other Limitations

- The DHPL_CALL_CBLAS option is not allowed when building the hybrid version of MP LINPACK.
- On Intel® 64 architecture processors user programs compiled with the GNU Fortran compiler (version 3.2.3) will likely get incorrect results from those functions in Intel® MKL that return single precision values, if -fno-f2c GNU Fortran compiler flag isn't used. The GNU Fortran compiler by default expects REAL*4 values in the first 8 bytes of the return register (just as a double precision value would be represented) while the Intel® Fortran compiler expects REAL*4 values in the first 4 bytes of the return register. The behavior of Intel® MKL is compatible with that of the Intel Fortran compiler. GNU Fortran compiler behavior could be changed to be compatible with the Intel Fortran compiler by using the -fno-f2c flag.
- FFT and PDE Support functions cannot be called from Fortran-77. These components have Fortran-90/95 interface specifics (structures, ..) that cannot be used with Fortran-77.
- We recommend that -Od be used when compiling test source code available with Intel® MKL. Current build scripts do not specify this option and default behavior for the compilers has changed to provide vectorization.
- All VSL functions return an error status, i.e., default VSL API is a function style now rather than a subroutine style used in earlier Intel® MKL versions. This means that Fortran users should call VSL routines as functions. For example:
  ```
    errstatus = vslrnggaussian(method, stream, n, r, a, sigma)
  ```
  rather than subroutines:
  ```
    call vslrnggaussian(method, stream, n, r, a, sigma)
  ```
  Nevertheless, Intel® MKL provides a subroutine-style interface for backward compatibility. To use subroutine-style interface, manually include mkl_vsl_subroutine.fi file instead of mkl_vsl.fi by changing the line include 'mkl_vsl.fi' mkl.fi (in the include directory) with the line include 'mkl_vsl_subroutine.fi'. VSL API changes don't affect C/C++ users.

## 6.3 Memory Allocation

In order to achieve better performance, memory allocated by Intel® MKL is not released. This behavior is by design and is a onetime occurrence for Intel® MKL routines that require workspace memory buffers. Even so, the user should be aware that some tools may report this as a memory leak. Should the user wish, memory can be released by the user program through use of a function (MKL_FreeBuffers()) made available in Intel® MKL or memory can be released after each call by setting the environment variable MKL_DISABLE_FAST_MM (see User's Guide in the doc directory for more details). Using one of these methods to release memory will not necessarily stop programs from reporting memory leaks, and in fact may increase the number of such reports should you make multiple calls to the library thereby requiring new allocations with each call. Memory not released by one of the methods described will be released by the system when the program ends. To avoid this restriction disable memory management as described above.

On Red Hat* Enterprise Linux 3.0, in order to ensure that the correct support libraries are linked, the environment variable LD_ASSUME_KERNEL must be set. For example: 'export LD_ASSUME_KERNEL=2.4.1'

## 6.4 Other Notes

The GMP component is located in the solver library. For Intel® 64 and IA-64 platforms these components support only LP64 interface.

# 7 Intel® Threading Building Blocks

This section summarizes changes, new features and late-breaking news about Intel® Threading Building Blocks (Intel® TBB).

- Unhandled exceptions in the user code executed in the context of TBB algorithms or containers may lead to segmentation faults when Intel(R) C++ Compiler 10.x is used with glibc 2.3.2, 2.3.3, or 2.3.4.
- To allow more accurate results to be obtained with Intel® Thread Checker or Intel® Thread Profiler, download the latest update releases of these products before using them with Intel® Threading Building Blocks.
- If you are using Intel® Threading Building Blocks and OpenMP* constructs mixed together in rapid succession in the same program, and you are using Intel compilers for your OpenMP* code, set KMP_BLOCKTIME to a small value (e.g., 20 milliseconds to improve performance. This setting can also be made within your OpenMP* code via the kmp_set_blocktime() library call. See the compiler OpenMP* documentation for more details on KMP_BLOCKTIME and kmp_set_blocktime().
- In general, non-debug ("release") builds of applications or examples should link against the non-debug versions of the Intel® Threading Building Blocks libraries, and debug builds should link against the debug versions of these libraries. See the Tutorial in the product documentation sub-directory for more details on debug vs. release libraries.

- When using Ubuntu* 7.04 in 64-bit mode, compilations can fail with error messages saying that "`::system' has not been declared". These failures can be worked around by removing libpthread-dev from the system. See the following link for more details: https://bugs.launchpad.net/ubuntu/+source/gcc-4.1/+bug/77559

## 8 Disclaimer and Legal Information