

Intel® Fortran Compiler Professional Edition 11.0 for Linux*

Installation Guide and Release Notes
23 September 2008

1 Introduction

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

1.1 Product Contents

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

- Intel® Fortran 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® Math Kernel Library
- Cluster OpenMP* (separate license required)
- On-disk documentation

1.2 System Requirements

1.2.1 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.

1.2.2 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.

1.2.3 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)
- 1.5GB 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

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)
- 1.5GB 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).
- 1.5GB 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)

Additional Requirements to use Intel® Cluster OpenMP* (optional feature)

- Minimum Hardware Required (per node)
 - Intel® 64 Architecture system or IA-64 Architecture system
- Recommended Hardware (per node)
 - 2 GB of RAM

- 10 GB of disk space
- Operating Systems:
 - Red Hat* Enterprise Linux* 3.0, 4.0
 - SUSE Linux Enterprise Server* 9.0, 10.0
- POSIX* threads: NPTL

For Infiniband* support:

- Open Fabrics Enterprise Distribution (OFED) 1.0 or later

The OFED software may be downloaded from <https://svn.openfabrics.org/svn/openib/gen2/branches/>

Recommended Software

- Intel® Trace Analyzer and Collector
- Intel® Thread Profiler
- Intel® Thread Checker

Cluster Requirements

One of the following supported communications fabrics: Ethernet*, Gigabit Ethernet*, Infiniband*, or any fabric that supports TCP/IP.

All nodes that will cooperate in the execution of a Cluster OpenMP* program must be running the same operating system and the same kernel version. The nodes should be as identical as possible with respect to mounted file systems and system paths.

There must be enough swap space available on disk for the program. In addition to the normal swap space needed by a Linux program, Cluster OpenMP* requires disk space allocated separately for sharable backing store. Sharable backing store is allocated in /tmp by default, and requires space equal to the size of twice the sharable pages allocated to the program. Use the --backing-store option in the kmp_cluster.ini file to allocate sharable backing store in some directory other than /tmp.

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 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.

1.3.1 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 Fortran 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.

1.4 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`
 - `lib`
 - `ia32`
 - `intel64`
 - `ia64`
 - `idb`
 - `mkl`
 - `Documentation`
 - `compiler_f`
 - `idb`
 - `mkl`
 - `man`
 - `Samples`

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.

1.5 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 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_cprof.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 also have the same-numbered version of Intel® C++ Compiler installed, it may also be removed.

1.6 Documentation

Product documentation can be found in the `Documentation` folder as shown under [Installation Folders](#).

1.7 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/flin>.

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

2 Intel® Fortran Compiler

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

2.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler for Linux* (8.0 and later) may be used in a build with version 11.0. Exceptions include:

- Objects built with the multi-file interprocedural optimization (`-ipo`) option must be recompiled.
- Objects built for the Intel® 64 architecture with a compiler version earlier than 10.0 and that use the `REAL(16)` or `REAL*16` datatypes must be recompiled.
- Objects built for the Intel® 64 or IA-64 architectures with a compiler version earlier than 10.0 and that have module variables must be recompiled. If non-Fortran sources reference these variables, the external names may need to be changed to remove an incorrect leading underscore.
- Modules that specified an `ATTRIBUTES ALIGN` directive and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.

Note: 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

- Features from Fortran 2003
 - Enumerators
 - Type extension (not polymorphic)
 - Allocatable scalar variables (not deferred-length character)
 - ERRMSG keyword for ALLOCATE and DEALLOCATE
 - SOURCE= keyword for ALLOCATE
 - MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC with CHARACTER arguments
 - Intrinsic modules IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES
 - ASSOCIATE construct
 - PROCEDURE declaration
 - Procedure pointers
 - ABSTRACT INTERFACE
 - PASS and NOPASS attributes
 - Structure constructors with component names and default initialization
 - Array constructors with type and character length specifications
 - BLANK, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
 - DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
- OpenMP* 3.0 support
- UNROLL_AND_JAM and NOUNROLL_AND_JAM directives
- VECTOR NONTEMPORAL directive now allows variables to be specified
- VECTOR TEMPORAL directive

2.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- -axSSE2
- -axSSE3
- -axSSSE3
- -axSSE4.1
- -diag-error-limit
- -diag-once
- -falign-stack
- -fast-transcendentals
- -finline
- -fma
- -fp-relaxed

- -fpie
- -fstack-protector
- -m32
- -m64
- -mia32
- -minstruction
- -mssse3
- -msse4.1
- -openmp-link
- -openmp-threadprivate
- -opt-block-factor
- -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-dir
- -prof-src-root
- -prof-src-root-cwd
- -tcollect-filter
- -vec
- -xHost
- -xSSE2
- -xSSE3
- -xSSE3_ATOM
- -xSSSE3
- -xSSE4.1

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	-xSSE4.1, -xSSE3, -xSSE3 or -xSSE2 as appropriate
Older Intel processor	-mia32
Non-Intel processor	-mSSE3, -mSSE2 or -mia32 as appropriate, based on CPUID flags

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 and Notes

2.4.1 Establishing the Compiler Environment

The `ifortvars.sh` (`ifortvars.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 `fc` or `fce` 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/uuu/bin/ifortvars.sh argument
```

Where `<install-dir>` is the installation directory (default for system-wide installation is `/opt/intel`) and `uuu` 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 gcc* 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 Procedure Pointers and the PASS Attribute

The version 11.0 compiler supports the Fortran 2003 Procedure Pointer feature and procedure pointers may appear as components of derived types. The PASS and NOPASS attributes specify whether or not the enclosing derived type is passed as an argument to the called procedure: the default is PASS. The standard also requires that procedure pointers with the PASS attribute be polymorphic. The version 11.0 compiler does not yet support polymorphic procedures, so while it is possible to write a program using procedure pointers that have the PASS attribute, it is not yet possible to do so in a standard-conforming manner.

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 Known Issues

2.5.1 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.

2.6 Fortran 2003 Feature Summary

The Intel Fortran Compiler supports many features that are new to the latest revision of the Fortran standard, Fortran 2003. Additional Fortran 2003 features will appear in future versions. Fortran 2003 features supported by the current compiler include:

- The Fortran character set has been extended to contain the 8-bit ASCII characters `~ \ [] ` ^ { } | # @`
- Names of length up to 63 characters
- Statements of up to 256 lines
- Square brackets `[]` are permitted to delimit array constructors instead of `(/)`
- Structure constructors with component names and default initialization
- Array constructors with type and character length specifications

- A named PARAMETER constant may be part of a complex constant
- Enumerators
- Allocatable components of derived types
- Allocatable scalar variables (not deferred-length character)
- ERRMSG keyword for ALLOCATE and DEALLOCATE
- SOURCE= keyword for ALLOCATE
- Type extension (not polymorphic)
- ASYNCHRONOUS attribute and statement
- BIND(C) attribute and statement
- PROTECTED attribute and statement
- VALUE attribute and statement
- VOLATILE attribute and statement
- INTENT attribute for pointer objects
- Reallocation of allocatable variables on the left hand side of an assignment statement when the right hand side differs in shape or length (requires option "assume realloc_lhs")
- ASSOCIATE construct
- In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=
- FLUSH statement
- WAIT statement
- ACCESS='STREAM' keyword for OPEN
- ASYNCHRONOUS keyword for OPEN and data transfer statements
- ID keyword for INQUIRE and data transfer statements
- POS keyword for data transfer statements
- PENDING keyword for INQUIRE
- The following OPEN numeric values can be of any kind: RECL=
- The following READ and WRITE numeric values can be of any kind: REC=, SIZE=
- The following INQUIRE numeric values can be of any kind: NEXTREC=, NUMBER=, RECL=, SIZE=
- Recursive I/O is allowed in the case where the new I/O being started is internal I/O that does not modify any internal file other than its own
- IEEE Infinities and NaNs are displayed by formatted output as specified by Fortran 2003
- BLANK, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
- DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
- In an I/O format, the comma after a P edit descriptor is optional when followed by a repeat specifier
- Rename of user-defined operators in USE
- INTRINSIC and NON_INTRINSIC keywords in USE
- IMPORT statement
- Allocatable dummy arguments
- Allocatable function results
- PROCEDURE declaration
- Procedure pointers

- ABSTRACT INTERFACE
- PASS and NOPASS attributes
- COMMAND_ARGUMENT_COUNT intrinsic
- GET_COMMAND intrinsic
- GET_COMMAND_ARGUMENT intrinsic
- GET_ENVIRONMENT_VARIABLE intrinsic
- IS_IOSTAT_END intrinsic
- IS_IOSTAT_EOR intrinsic
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC intrinsics allow CHARACTER arguments
- MOVE_ALLOC intrinsic
- NEW_LINE intrinsic
- SELECTED_CHAR_KIND intrinsic
- The following intrinsics take an optional KIND= argument: ACHAR, COUNT, IACHAR, ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE, UBOUND, VERIFY
- ISO_C_BINDING intrinsic module
- IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES intrinsic modules
- ISO_FORTRAN_ENV intrinsic module

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, 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
- FORTRAN improvements
- 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:

Intel® Fortran Compiler Professional Edition
11.0 for Linux* Installation Guide and Release Notes

Error: could not start debuggee

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.f90":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 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.

4 Intel® Math Kernel Library

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

Intel® Fortran Compiler Professional Edition
11.0 for Linux* Installation Guide and Release Notes

4.1 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.

4.2 Known Limitations

4.2.1 Limitations to the sparse solver and optimization solvers:

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

4.2.2 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`

4.2.3 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.

4.2.4 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

4.2.5 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

4.2.6 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.

4.2.7 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).

4.2.8 Limitations to the g77 compiler support:

- Some Intel® MKL functions contain underscore in their names (i.e. mkl_dcsrsmv, mkl_cspblas_dcsrsmv) 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

4.2.9 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.

4.3 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'

4.4 Other Notes

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

5 Cluster OpenMP*

5.1 Overview

Cluster OpenMP* is a system that supports running an OpenMP* program on a set of nodes connected by a communication fabric, such as Ethernet*. Such nodes do not have the shared memory hardware that OpenMP* is designed for, so Cluster OpenMP* simulates that hardware with a software mechanism. The software mechanism used by Cluster OpenMP* is commonly referred to as a distributed shared memory system.

Cluster OpenMP* is an unsupported add-on to the Intel Compiler. You must obtain a no-charge separate license in order to compile your code for Cluster OpenMP*. For more information, and to request help for using Cluster OpenMP, please visit <http://whatif.intel.com>.

5.2 Product Contents

Cluster OpenMP* consists of two major parts: the OpenMP* support layer and the DVSM layer. The OpenMP* support layer implements the semantics of OpenMP* and uses the DVSM layer to manage the memory shared across the cluster. The DVSM layer is an exclusively-licensed and heavily modified extension of the TreadMarks distributed shared memory system developed at Rice University by Prof. Willy Zwaenepoel, Prof. Alan Cox, their colleagues and students.

5.3 New Features

Please see section 1.4 in the Cluster OpenMP* User Manual for details of the following items.

- Reduction code speed and scalability improvement
- New top-level cluster_omp directory in the compiler directory tree
- New performance analysis tools: segvprof.pl and dashboard

- Sharable sections functionality

5.4 Known Issues and Limitations

For Intel® 64 architecture only, the following issue exists:

- Older versions of the Cluster OpenMP* runtime library (libclusterguide.so) use some symbols from the library "libirc" that cause incompatibilities with newer hardware. Newer versions of libclusterguide.so no longer use libirc. With older versions of libclusterguide.so, at runtime you may see an error similar to the following:

```
a.out: symbol lookup error: a.out: undefined symbol:
__intel_cpu_indicator
```

If you receive this error, you must relink your application with a newer libclusterguide.so library.

For all architectures, the following issues exist:

- Cluster OpenMP* programs built with C/C++ compilers version 9.1.045 and earlier or Fortran compilers version 9.1.040 and earlier must be re-linked or re-compiled with later versions of the compiler before they can be executed with the associated later versions of the Cluster OpenMP* libraries.
- Cluster OpenMP* does not support the version of Posix threads known as "linuxthreads". It does support the version of Posix threads known as "NPTL". You can find out which version of Posix threads your kernel supports with the following Linux command:

```
$ getconf GNU_LIBPTHREAD_VERSION
```

The output will look something like either this:

```
NPTL 0.60
```

or this:

```
linuxthreads-0.10
```

If your system supports linuxthreads, then contact your system administrator to get NPTL support enabled.

- Any system that has a file /etc/security/limits.conf must have limits set in that file for "memlock" that are at least as large as the following:
 - soft memlock 4000000
 - hard memlock 4000000
- Some recent Linux distributions, particularly SuSE 10, have enabled a kernel security feature known as "randomize_va_space". This feature causes the virtual addresses of memory mapped code and data to change at every process invocation. This feature causes problems with Cluster OpenMP*, since Cluster OpenMP* requires every process

to map sharable data at the same virtual address. You may determine whether your system is affected by this by looking at the file `/proc/sys/kernel/randomize_va_space`. If it contains "1", then this feature is enabled on your system. To make Cluster OpenMP* work properly in this situation, you must disable `randomize_va_space` by putting a "0" in that file. To do this, you should login as "root" and edit the file `/etc/sysctl.conf` by adding the line:

```
kernel.randomize_va_space=0
```

then, reboot.

- Cluster OpenMP* is only tested and supported in configurations where the nodes that will cooperate in the execution of a Cluster OpenMP* program are all running the same operating system and the same kernel version.
- Cluster OpenMP* supports only one level of nested parallelism. When an outer parallel region is active and an inner parallel region is encountered, the inner region is serialized.
- The limit on the total number of threads supported by Cluster OpenMP*, in all processes, is nearly 65,536.
- The limit on the total amount of sharable memory supported by Cluster OpenMP* is 256 GBytes.
- If you are trying to use a library statically linked with OpenMP*, while building a Cluster OpenMP* program, you must make sure that the Cluster OpenMP* library (`libclusterguide.so`) is linked before the library containing statically-linked OpenMP code. See section 10.2 of the Cluster OpenMP* User's Guide for more information.
- An OpenMP* parallel region not nested within a Cluster OpenMP* parallel region will execute in parallel on a single node. An OpenMP* parallel region nested within a Cluster OpenMP* parallel region will execute on each thread that encounters it.
- A Cluster OpenMP* program does not currently work as part of a shared object library, since it depends on each sharable variable being loaded at the same virtual address on all nodes.
- Some versions of the C++ Standard Template Library seem to have bugs related to the use of sharable allocators. See sections 10.7.1.4 and 10.7.1.5 of the Cluster OpenMP* User's Guide for more information on using sharable allocators with STL containers.

5.5 Documentation and Samples

Support materials are available in the installed compiler directory structure. See the directories:

`<install-dir>/Documentation/cluster_omp/docs` - for Cluster OpenMP* documentation

`<install-dir>/Documentation/cluster_omp/examples` - for Cluster OpenMP* usage examples

`<install-dir>/Documentation/cluster_omp/tools` - for tools useful with Cluster OpenMP* programs

6 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Celeron, Centrino, Intel, Intel logo, Intel386, Intel486, Intel Atom, Intel Core, Itanium, MMX, Pentium, VTune, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.

Copyright © 2008, Intel Corporation. All Rights Reserved.