

# Intel® Fortran Compiler Professional Edition 11.1 for Linux\* Installation Guide and Release Notes

---

Document number: 321415-002US  
23 April 2009

## Table of Contents

- 1 Introduction ..... 3
  - 1.1 Product Contents ..... 3
  - 1.2 System Requirements..... 3
    - 1.2.1 Red Hat Enterprise Linux\* 3, SUSE LINUX Enterprise Server\* 9 Support  
Deprecated ..... 5
  - 1.3 Documentation..... 5
  - 1.4 Technical Support..... 6
- 2 Installation..... 6
  - 2.1 Silent Install ..... 6
  - 2.2 Known Installation Issues..... 6
  - 2.3 Installation Folders..... 7
  - 2.4 Removal/Uninstall ..... 8
- 3 Intel® Fortran Compiler ..... 8
  - 3.1 Compatibility ..... 9
  - 3.2 New and Changed Features ..... 9
    - 3.2.1 Features from Fortran 2003 ..... 9
    - 3.2.2 Other Changes ..... 10
  - 3.3 New and Changed Compiler Options ..... 10
  - 3.4 Other Changes and Notes ..... 10
    - 3.4.1 Optimization Reports Disabled by Default..... 10
    - 3.4.2 Establishing the Compiler Environment..... 11
    - 3.4.3 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2  
(Intel® SSE2)..... 11
    - 3.4.4 OpenMP\* Libraries Default to “compat”..... 11

3.4.5	OpenMP* Libraries Default to Dynamic Linking.....	11
3.4.6	Sampling-based Profile Guided Optimization Feature Removed.....	11
3.5	Known Issues .....	12
3.5.1	The behavior default behavior for <code>KMP_AFFINITY</code> has changed.....	12
3.5.2	Fatal error from old version of <code>ld</code> .....	12
3.5.3	Limited Support for Empty Derived Types.....	12
3.6	Fortran 2003 Feature Summary .....	13
4	Intel® Debugger (IDB) .....	16
4.1	Setting up the Java Runtime Environment .....	16
4.2	Starting the Debugger.....	16
4.3	Additional Documentation .....	16
4.4	Debugger Features.....	17
4.4.1	Main Features of IDB.....	17
4.4.2	New and Changed Features .....	17
4.5	Known Problems.....	17
4.5.1	Signals Dialog not working.....	17
4.5.2	Debugging Shared Libraries .....	17
4.5.3	<code>list</code> command .....	17
4.5.4	Resizing GUI.....	18
4.5.5	Kill Process.....	18
4.5.6	OpenMP Locks: "No information available".....	18
4.5.7	Online Help Error "Unable to open web browser" .....	18
5	Intel® Math Kernel Library .....	18
5.1	New and Changed Features .....	18
5.2	Known Limitations.....	20
5.2.1	Limitations to the sparse solver and optimization solvers:.....	20
5.2.2	Limitations to the FFT functions: .....	20
5.2.3	Limitations to the LAPACK functions:.....	21
5.2.4	Limitations to the Vector Math Library (VML) and Vector Statistical Library (VSL) functions: .....	21
5.2.5	Limitations to the ScaLAPACK functions:.....	21
5.2.6	Limitations to the ILP64 version of Intel® MKL:.....	21
5.2.7	Limitations to the Fortran 95 interface to LAPACK: .....	21

5.2.8	Limitations to the g77 compiler support:.....	21
5.2.9	Other Limitations.....	21
5.3	Memory Allocation .....	22
5.4	Other Notes .....	23
6	Disclaimer and Legal Information.....	23

## 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 11.1 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
- On-disk documentation

### 1.2 System Requirements

For an explanation of architecture names, see <http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/>

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

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium 4 processor or later, or compatible non-Intel processor)
  - Development for a target different from the host may require optional library components to be installed from your Linux Distribution.
  - For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB 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\* 10

- Red Hat Enterprise Linux\* 3, 4, 5
- SUSE LINUX Enterprise Server\* 9, 10, 11
- TurboLinux\* 11
- Ubuntu\* 8.10
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- Linux component compat-libstdc++ providing libstdc++.so.5
- If developing on an Intel® 64 architecture system, Linux component glibc-devel.i386 providing crt1.o

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

- A PC based on an Intel® 64 architecture processor (Intel® Pentium 4 processor or later, or compatible non-Intel processor)
  - For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB 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\* 10
  - Red Hat Enterprise Linux\* 3, 4, 5
  - SGI ProPack\* 5
  - SUSE LINUX Enterprise Server\* 9, 10, 11
  - TurboLinux\* 11
  - Ubuntu\* 8.10
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- 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 IA-64 architecture processor (Intel® Itanium®)
- 1GB of RAM (2 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, 11
- TurboLinux\* 11
- Ubuntu\* 8.10
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- 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 (JRE) 5.0 (also called 1.5.0) or 6.0
  - A 32-bit JRE must be used on an IA-32 architecture system and a 64-bit JRE must be used on an Intel® 64 architecture system

### **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.

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

In a future major release of Intel Fortran 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.

### **1.3 Documentation**

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

## 1.4 Technical Support

Register your license at the [Intel® Software Development Products Registration Center](https://www.intel.com/software/products/registration-center). Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit:

<http://www.intel.com/software/products/support/>

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

## 2 Installation

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.

### 2.1 Silent Install

For information on automated or “silent” install capability, please see <http://software.intel.com/en-us/articles/intel-compilers-for-linux-silent-installation-guide/>

### 2.2 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 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.

- The Intel Fortran Compiler Professional 11.1 product is fully supported on Ubuntu 8.10 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 8.10. 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 Fortran Compiler Professional 11.1 product on an Intel® 64 architecture Ubuntu system, please visit the Intel Software Evaluation Center (<http://www.intel.com/cd/software/products/asm-na/eng/download/eval/>) to obtain an evaluation serial number.

## 2.3 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- <install-dir>/Compiler/11.1/xxx/
  - o bin
    - ia32
    - intel64
    - ia64
  - o include
    - ia32
    - intel64
    - ia64
  - o lib
    - ia32
    - intel64
    - ia64
  - o idb
    - gui
    - ia32
    - ia64
    - intel64

- lib
  - third\_party
- o mkl
  - benchmarks
  - examples
  - include
  - interfaces
  - lib
  - tests
  - tools
- o Documentation
- o man
- o 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.

## 2.4 Removal/Uninstall

Removing (uninstalling) the product should be done by the same user who installed it (root or a non-root user). If `sudo` was used to install, it must be used to uninstall as well. 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.

## 3 Intel® Fortran Compiler

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

## 3.1 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 11.1. 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.

## 3.2 New and Changed Features

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard ([http://j3-fortran.org/doc/2003\\_Committee\\_Draft/04-007.pdf](http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf)) if necessary.

### 3.2.1 Features from Fortran 2003

- Object-oriented features
  - CLASS declaration
  - SELECT TYPE construct
  - EXTENDS\_TYPE\_OF and SAME\_TYPE\_AS intrinsic functions
  - Polymorphic entities
  - Inheritance association
  - Deferred bindings and abstract types
  - Type inquiry intrinsic functions
- Type-bound procedures
  - TYPE CONTAINS declaration
  - ABSTRACT attribute
  - DEFERRED attribute
  - NON\_OVERRIDABLE attribute
  - **Note:** GENERIC attribute and type-bound operators are not supported in this release
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- NAMELIST I/O is permitted on an internal file

- Restrictions on entities in a NAMELIST group are relaxed
- Changes to how IEEE Infinity and NaN is represented in formatted input and output
- The COUNT\_RATE argument to the SYSTEM\_CLOCK intrinsic may be a REAL of any kind
- Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
- MAXLOC or MINLOC of a zero-sized array returns zero if the option `-assume noold_maxminloc` is specified. Fortran 95 specified that the value was processor-dependent and Intel Fortran returned 1. Performance will be lower if `-assume noold_maxminloc` is specified.

### 3.2.2 Other Changes

- When string length checking is in effect (`-check bounds`), and a character object is passed as an argument, the minimum of the passed length and the declared length in the called procedure is used as an upper limit
- Input value items in the form of a LOGICAL constant, for example T or .F, are no longer accepted during list-directed or namelist-directed input when the corresponding variable in the I/O list is not LOGICAL. The new `-assume old_logical_ldio` option can be used to restore the older behavior.
- Per-compilation control of floating point exception behavior (`-fpe-all`)

## 3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- `-assume [no]ieee_fpe_flags`
- `-assume [no]old_logical_ldio`
- `-assume [no]old_maxminloc`
- `-diag-enable:sc-include`
- `-diag-enable:sc-parallel`
- `-fpe-all`

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

## 3.4 Other Changes and Notes

### 3.4.1 Optimization Reports Disabled by Default

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

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

### 3.4.2 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.x, 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/11.1/xxx/bin/ifortvars.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.

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

When compiling for the IA-32 architecture, `-msse2` (formerly `-xW`) is the default as of version 11.0. 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`

### 3.4.4 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.x, 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.

### 3.4.5 OpenMP\* Libraries Default to Dynamic Linking

As of version 11.0, OpenMP applications link to the dynamic OpenMP libraries by default. To specify static linking of the OpenMP libraries, specify `-openmp-link static` .

### 3.4.6 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.

## 3.5 Known Issues

### 3.5.1 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.x 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.x compilers (`KMP_AFFINITY=disabled`). Setting `KMP_AFFINITY=disabled` will prevent the OpenMP runtime library from making any affinity-related system calls.

### 3.5.2 Fatal error from old version of `ld`

In some circumstances, linking of an application with the version 11.x compiler will fail with an `ld` internal error similar to the following:

```
ld: BFD 2.15.92.0.2 20040927 internal error, aborting at
../bfd/reloc.c line 444 in bfd_get_reloc_size
ld: Please report this bug.
```

To resolve this, install a more recent version of `binutils`. 2.17.50 is the recommended minimum version.

### 3.5.3 Limited Support for Empty Derived Types

Fortran 2003 adds the ability to declare a derived type with no data components. The Intel compiler has limited support for these in the current release. These limitations will be lifted in a future release of the compiler. The limitations are as follows:

- When an object of derived type is declared, the type must have at least one data component. Extending an empty type is supported. For example:

```
type t
end type
```

```
type, extends (t) :: t1
end type
```

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

```
type, extends (t2) :: t3
end type
```

```
type (t) :: recl ! Not supported, type t is empty
```

```

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

```

An exception is that it is supported to declare a class object with an empty type, for example:

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

If an unsupported use of an empty type is seen, the compiler will issue the diagnostic:

```
Declaring an object with no data component fields is not yet supported
```

- Referencing a component that is an empty type is not supported. For example, assuming the declarations above, in:

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

the references to `rec4%t3`, `rec4%t1`, `rec4%t`, `rec3%t1`, and `rec3%t` are not supported. References to `rec3%t2` and `rec4%t2` will be supported. If an unsupported reference is seen, the compiler will issue the diagnostic:

```
Accessing an empty type is not yet supported
```

- A type constructor for an empty type is not supported. Again assuming the declarations above, the type constructor `t()` is not supported. If an unsupported constructor is seen, the compiler will issue the diagnostic:

```
A type constructor for an empty type is not yet supported
```

### 3.6 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
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- ERRMSG keyword for ALLOCATE and DEALLOCATE
- SOURCE= keyword for ALLOCATE
- Type extension
- CLASS declaration
- Polymorphic entities
- Inheritance association
- Deferred bindings and abstract types
- Type-bound procedures
- TYPE CONTAINS declaration
- ABSTRACT attribute
- DEFERRED attribute
- NON\_OVERRIDABLE attribute
- 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
- SELECT TYPE construct
- In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=
- NAMELIST I/O is permitted on an internal file
- Restrictions on entities in a NAMELIST group are relaxed
- Changes to how IEEE Infinity and NaN is represented in formatted input and output
- 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, DECIMAL, 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
- The COUNT\_RATE argument to the SYSTEM\_CLOCK intrinsic may be a REAL of any kind
- Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
- MAXLOC or MINLOC of a zero-sized array returns zero if the option `-assume noold_maxminloc` is specified.
- Type inquiry intrinsic functions
- COMMAND\_ARGUMENT\_COUNT intrinsic
- EXTENDS\_TYPE\_OF and SAME\_TYPE\_AS intrinsic functions
- 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

Fortran 2003 features not yet supported include:

- Type-bound operators and the GENERIC binding for type-bound procedures
- User-defined derived type I/O
- Parameterized derived types

## 4 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.

### 4.1 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) or version 6.0 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
```

### 4.2 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>`

### 4.3 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.

## 4.4 Debugger Features

### 4.4.1 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.

### 4.4.2 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

## 4.5 Known Problems

### 4.5.1 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.

### 4.5.2 Debugging Shared Libraries

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

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.

### 4.5.3 `list` command

In GDB mode, unquoted filenames do not work. The workaround is to use quoted filenames, e.g. `list "test.f90":10`

#### 4.5.4 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.

#### 4.5.5 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.

#### 4.5.6 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.

#### 4.5.7 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.

## 5 Intel® Math Kernel Library

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

### 5.1 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 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.

## 5.2 Known Limitations

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

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

### 5.2.2 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: `-WI,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -WI,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread`

### 5.2.3 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.

### 5.2.4 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

### 5.2.5 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

### 5.2.6 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.

### 5.2.7 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).

### 5.2.8 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`

### 5.2.9 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 `-O3` 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 = vslnrggaussian(method, stream, n, r, a, sigma)`  
rather than subroutines:  
    `call vslnrggaussian(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.

### 5.3 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'`

## 5.4 Other Notes

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

## 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.

Parts of this product were built using third party libraries. Pursuant to the licenses ruling these libraries, Intel makes them available to users of this product. The libraries can be downloaded from Intel Software Development Products Knowledge Base article <http://software.intel.com/en-us/articles/open-source-downloads/> . Please note that download of these libraries is not required to use the product.

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 © 2009 Intel Corporation. All Rights Reserved.