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

Document number: 321415-002US
3 December 2009

Table of Contents

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 Change History

This section highlights important changes in product updates. For a list of corrections to reported problems, please read Intel® Professional Edition Compilers 11.1 Fixes List for the compiler and Intel® Math Kernel Library 10.2 Fixes List for the Intel® Math Kernel Library.

Update 4

- Intel® Math Kernel Library updated to 10.2 Update 3
- Corrections to reported problems

Update 3 (11.1.059)

- Corrections to reported problems

Update 2 (11.1.056)

- Support for Ubuntu* 9.04 added
- Note added about non-RPM installation forced on Fedora* 10 systems.
- Hidden arguments are no longer used with BIND(C) routines. This may require source changes.
- Added mention of new compiler options `-mkl` and `-xAVX`
- Note added about change in behavior for `-warn interface`
- Corrections to reported problems

Update 1 (11.1.046)

- Sources declaring or using derived types containing type-bound procedures must be recompiled
- Note added about change in behavior of `-O0`

- FORT_BLOCKSIZE and FORT_BUFFERCOUNT environment variables documented
- Corrections to reported problems

1.2 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 10.2 Update 3
- On-disk documentation

1.3 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* 9.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- Linux component compat-libstdc++ providing libstdc++.so.5
- Library libunwind.so is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.
- If developing on an Intel® 64 architecture system, some Linux distributions may require installation of one or more of the following additional Linux components: ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib

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
    - SUSE LINUX Enterprise Server* 9, 10, 11
    - TurboLinux* 11
    - Ubuntu* 9.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- Linux component compat-libstdc++ providing libstdc++.so.5
- Library libunwind.so is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.
- 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* 9.04
- 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 (JRE) 5.0 (also called 1.5) or 6.0 (1.6)
    - 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.

1.3.1 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.

1.4 Documentation

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

1.5 Japanese Language Support

Intel compilers provide support for Japanese language users. Error messages, visual development environment dialogs and some documentation are provided in Japanese in addition to English. By default, the language of error messages and dialogs matches that of your operating system language selection. Japanese-language documentation can be found in the `ja_JP` subdirectory for documentation and samples.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions at http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-linux/

1.6 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/

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-version-111-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.
- On Fedora* 10 systems, the compiler forces a non-RPM install because some versions of Fedora 10 contain a defective `rpm` utility that prevents the Intel® compiler from installing properly.
- The version 11.1 product is fully supported on Ubuntu 9.04 IA-32 and Intel® 64 architecture systems. Due to a restriction in the licensing software, however, it is not possible to use the Trial License feature when evaluating IA-32 components on an Intel® 64 architecture system with Ubuntu 9.04. Earlier versions of Ubuntu, not officially supported by this release of software, may have similar problems. This affects using a Trial License only. Use of serial numbers, license files, floating licenses or other license manager operations, and off-line activation (with serial numbers) is not affected. If you need to evaluate IA-32 components of the version 11.1 product on an Intel® 64 architecture Ubuntu system, please visit the Intel Software Evaluation Center (http://www.intel.com/cd/software/products/asmo-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`*`/`
  - `bin`
    - `ia32`
    - `intel64`
    - `ia64`
  - `include`
    - `ia32`
    - `intel64`
    - `ia64`
  - `lib`
    - `ia32`
    - `intel64`
    - `ia64`
  - `idb`

        - gui
        - ia32
        - ia64
        - intel64
        - lib
        - third_party
    - mkl
        - benchmarks
        - examples
        - include
        - interfaces
        - lib
        - tests
        - tools
    - Documentation
    - man
    - Samples

Where <install-dir> is the installation directory (default for system-wide installation is /opt/intel) and *xxx* is the three-digit build 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.

3.1.1 Incorrect Derived Type Layout for Type-Bound Procedures

The initial version 11.1 compiler incorrectly adds unused space to a derived type containing type-bound procedures. This error was corrected in version 11.1 Update 1. All sources declaring or using objects of such type, and which were compiled with the initial release 11.1 compiler, must be recompiled with version 11.1 Update 1 or later.

3.1.2 Removal of Inappropriate Hidden Arguments for BIND(C) Procedures

Earlier versions of the Intel Fortran Compiler incorrectly passed or used hidden arguments for procedures with the BIND(C) attribute. For example, if a Fortran routine had a CHARACTER argument and had BIND(C) specified, the compiler would assume that two hidden arguments were passed for the function return address and length, plus another hidden argument for each character argument. While such hidden arguments are necessary and correct for non-interoperable (without the BIND(C) attribute) Fortran procedures, the Fortran standard prohibits them for interoperable procedures. The Fortran and C argument list must have a 1:1 correspondence.

In many cases where this problem occurs, the procedure does not meet the standard's rules for interoperable procedures. For example, if an argument is of type CHARACTER, the length must be 1 – an array of such characters may be an argument. Arrays of any nature are not permitted as function return values for interoperable procedures. However, it is possible to create a standard-conforming interoperable procedure for which the compiler was passing or expecting hidden arguments.

In 11.1 Update 2, the compiler has been corrected to no longer pass or expect hidden arguments. If you wrote C code which assumed that such hidden arguments needed to be

passed you will have to rewrite it and recompile. You may find that in some cases, functions will need to be converted to subroutines with the result variable passed as an actual argument. We apologize for the inconvenience, but it is important for correctness and portability to have this error repaired.

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) 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. Similarly, when the I/O list variable is of type LOGICAL, the corresponding input value must be in the form of a LOGICAL constant. 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
- -mkl[=lib]
- -xAVX

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

3.3.1 –O0 no longer implies –mp

In version 11.1, the `-O0` option for disabling optimizations no longer implies `-mp` for maximizing floating-point precision. The `-mp` switch is deprecated, so we recommend using an explicit `-fp-model` option for applications that are sensitive to floating-point precision changes.

3.3.2 `-warn interface` now implies `-gen-interface`

As of version 11.1, specifying `-warn interface` enables both the generation and use of interfaces for the purpose of error checking. You no longer need to also specify `-gen-interface` if `-warn interface` is in effect.

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 some non-Intel 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 New Environment Variables to Control I/O

Version 11.1 supports two additional environment variables which can be used to affect I/O behavior when an application is run.

`FORT_BLOCKSIZE` lets you specify the default `BLOCKSIZE` value to be used when `BLOCKSIZE=` is omitted on the `OPEN` statement. Valid sizes are 0 to 2147467264. Sizes will be rounded up to the nearest 512-byte boundary. The default `BLOCKSIZE` value is now 128KB.

`FORT_BUFFERCOUNT` lets you specify the default `BUFFERCOUNT` value to be used when `BUFFERCOUNT=` is omitted on the `OPEN` statement. Valid values are 0 to 127. If 0 is specified, the default value of 1 will be used.

3.4.5 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.

3.4.6 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 .`

3.4.7 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 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) :: rec1 ! 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 the 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 registers window
- OpenMP support
    - Information windows for Tasks, Barriers, Taskwaits, Locks, Teams and Task Spawn Tree
    - Data sharing events and reentrancy call detection
    - Debugging serialized code without recompilation
- Internationalization support

4.5 Known Problems

4.5.1 Data Sharing Detection Issues

- When the `Stop On Event` icon is deactivated, or the `Parallel > Stop on Event` menu item is unchecked, data sharing events are not collected in the Data Sharing Events window. If you stop the debugger and open the Data Sharing Events window, you will see only the last event.

- When the `Data Sharing Events` window is closed and reopened, a new Analysis Run Node will be displayed that duplicates recent events.
- When the `Data Sharing Events` window is closed during data sharing detection, only the last event is displayed when the window is reopened after detection.

4.5.2 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.

4.5.3 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.

4.5.4 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.

4.5.5 Decimal Floating Point Not Supported

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

4.5.6 `$cdir, $cwd` Directories

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

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

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

4.5.7 `info stack` Usage

The debugger command `info stack` does not currently support negative frame counts in the optional syntax below:

```
info stack [num]
```

A positive frame count num will print the innermost num frames. A negative or zero count will print no frames rather than the outermost num frames.

4.5.8 `$stepg0` Default Value Changed

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

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

4.5.9 SIGTRAP error on some Linux* Systems

On some rare cases with special Linux kernels a SIGTRAP error may occur when the debugger stops at a breakpoint and you continue debugging. As a workaround you can define the SIGTRAP signal as follows on command line:

```
(idb) handle SIGTRAP nopass noprint nostop
SIGTRAP is used by the debugger.
SIGTRAP       No        No      No              Trace/breakpoint trap
(idb)
```

4.5.10 idb GUI cannot be used to debug MPI processes

The idb GUI cannot be used to debug MPI processes. The command line interface (idbc) can be used for this purpose.

5 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about Intel® Math Kernel Library (Intel® MKL) 10.2 Update 3 as part of Intel® Fortran Compiler Professional Edition.

5.1 Changes in This Version

For further information on improvements in this and previous releases, see http://software.intel.com/en-us/articles/new-in-intel-mkl-10-2/

5.1.1 Performance Improvements

- BLAS
  - Threaded the 32-bit OS versions of the following BLAS Level 1 and 2 functions for Intel® Core™ i7 processors and Intel® Xeon® processor 5300, 5400, and 5500 series: (D,S,C,Z)COPY, (D,S,C,Z)SWAP, (D,S,C,Z)AXPY, (S,C)ROT, (S,C)DOT, CDOTC, (D,S,C,Z)GEMV, (D,S,C,Z)TRMV, (S,C)SYMV, (S,C)SYR, (S,C)SYR2
  - Improved 32-bit and 64-bit OS versions of the following BLAS level 1 functions for Intel® Xeon® processors 5300, 5400, 5500: ZAXPY, ZSCAL, ZDOT(U,C), and (D,S)ROT
  - Improved DGEMM threading efficiency for matrices with many more rows than columns for Intel® Xeon® processor 5300
- LAPACK
  - Improved scalability of the following LAPACK functions: ?POTRF, ?GEBRD, ?SYTRD, ?HETRD, and ?STEDC divide and conquer eigensolvers
- FFTs
  - Updated underlying kernels to provide widespread performance improvements in FFTs
  - Improved threading of 3D FFTs when a small number of transforms are calculated with a single function call

    - Extended threading to small size multidimensional transforms
- VML
    - Further optimization for these VML functions on Intel® Xeon® processor 5500 series: v(s,d)Asin, v(s,d)Acos, v(s,d)Ln, v(s,d)Log10, vsLog1p, v[s/d]Hypot
- VSL
    - Improved performance of viRngPoisson and viRngPoissonV random number generators

5.1.2 Usability and Interface Improvements

- Improved example programs for uBLAS, Java, FFTW3, LAPACK95, and BLAS95
- Some examples in the reference manual were removed where identical examples in source code form also appeared in the examples directory
- New 64-bit integer (ILP64) fftw_mpi interfaces for cluster FFTs

5.2 Known Issues

A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at http://software.intel.com/en-us/articles/known-limitations-in-intel-mkl-10-2

5.3 Notices

The following change is planned for future versions of Intel MKL. Please contact Technical Support if you have concerns:

- Content in the libraries containing `solver` in the filenames will be moved to the core library in a future version of Intel MKL. These `solver` libraries will then be removed.

5.4 Attributions

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

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

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

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

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

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

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.