

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

---

Document number: 321416-002US  
3 December 2009

## Table of Contents

|       |  |   |
|-------|--|---|
| 1     | Introduction .....   | 3 |
| 1.1   | Change History .....   | 3 |
| 1.2   | Product Contents .....                                       | 3 |
| 1.3   | System Requirements.....                                     | 4 |
| 1.4   | Documentation.....   | 4 |
| 1.5   | Technical Support.....                                       | 4 |
| 2     | Installation.....  | 4 |
| 2.1   | Installation Folders.....                                    | 5 |
| 2.2   | Relocating Product After Install .....                       | 5 |
| 2.3   | Removal/Uninstall .....                                      | 6 |
| 3     | Intel® Fortran Compiler.....                                 | 6 |
| 3.1   | Compatibility .....  | 6 |
| 3.1.1 | Incorrect Derived Type Layout for Type-Bound Procedures..... | 6 |
| 3.2   | New and Changed Features .....                               | 6 |
| 3.2.1 | Features from Fortran 2003 .....                             | 7 |
| 3.2.2 | Other Changes .....  | 7 |
| 3.3   | New and Changed Compiler Options.....                        | 8 |
| 3.3.1 | -O0 no longer implies -mp .....                              | 8 |
| 3.4   | Other Changes .....  | 8 |
| 3.4.1 | Optimization Reports Disabled by Default.....                | 8 |
| 3.4.2 | New Environment Variables to Control I/O.....                | 8 |
| 3.4.3 | Environment Setup Script Changed.....                        | 8 |
| 3.5   | Known Issues .....   | 9 |
| 3.5.1 | Limited Support for Empty Derived Types.....                 | 9 |

|        |   |    |
|--------|---|----|
| 3.6    | Fortran 2003 Feature Summary .....            | 10 |
| 4      | Intel® Debugger (IDB) .....                   | 13 |
| 4.1    | Known Problems.....                           | 13 |
| 4.1.1  | Dwarf vs. Stabs Debug Formats .....           | 13 |
| 4.1.2  | Compilation Requirements.....                 | 13 |
| 4.1.3  | Non-local Binary and Source File Access ..... | 13 |
| 4.1.4  | Local variables may not be visible.....       | 14 |
| 4.1.5  | Printing Fortran REAL*16 variables .....      | 14 |
| 4.1.6  | Debugging applications that fork.....         | 14 |
| 4.1.7  | Debugging applications that exec .....        | 14 |
| 4.1.8  | Fortran alternate entry points .....          | 14 |
| 4.1.9  | Snapshots.....                                | 14 |
| 4.1.10 | Debugging optimized code.....                 | 14 |
| 4.1.11 | Watchpoints.....                              | 14 |
| 4.1.12 | Fortran modules and commons .....             | 15 |
| 4.1.13 | Graphical User Interface (GUI) .....          | 15 |
| 4.1.14 | MPP Debugging Restrictions .....              | 15 |
| 4.1.15 | Function Breakpoints .....                    | 15 |
| 4.1.16 | Core File Debugging.....                      | 15 |
| 4.1.17 | Universal Binary Support .....                | 15 |
| 4.1.18 | Debugger variable \$threadlevel .....         | 16 |
| 4.1.19 | Open File Descriptors Limitation .....        | 16 |
| 4.1.20 | \$cdir, \$cwd Directories .....               | 16 |
| 4.1.21 | info stack Usage.....                         | 16 |
| 4.1.22 | \$stepg0 Default Value Changed.....           | 16 |
| 5      | Intel® Math Kernel Library .....              | 17 |
| 5.1    | Changes in This Version.....                  | 17 |
| 5.1.1  | New features.....                             | 17 |
| 5.1.2  | Usability/Interface improvements .....        | 17 |
| 5.1.3  | Performance improvements .....                | 18 |
| 5.2    | Known Issues .....                            | 19 |
| 5.3    | Notices.....                                  | 19 |
| 5.4    | Attributions.....                             | 19 |

## 1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and 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](#).

#### Update 4

- Corrections to reported problems

#### Update 3 (11.1.076)

- Corrections to reported problems

#### Update 2 (11.1.067)

- Added mention of [new compiler option `-mk1`](#)
- Mac OS\* X 10.6.1 “Snow Leopard” is now supported
- Corrections to reported problems

#### Update 1 (11.1.058)

- 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 Mac OS\* X* includes the following components:

- Intel® Fortran Compilers for building applications that run on Intel-based Mac\* systems running the Mac OS\* X operating system
- Intel® Debugger
- Intel® Math Kernel Library 10.2 Update 2
- Integration into the Xcode\* development environment (Limited Feature)
- On-disk documentation

### 1.3 System Requirements

- An Intel®-based Apple\* Mac\* system
- 1GB RAM minimum, 2GB RAM recommended
- 2GB free disk space
- Mac OS\* X 10.5.6 and Xcode\* 3.1.2, or Mac OS\* X 10.5.7 and Xcode\* 3.1.3, or Mac OS\* X 10.5.8 and Xcode\* 3.1.4, or Mac OS\* X 10.6.1 and Xcode\* 3.2, or Mac OS\* 10.6.2 and Xcode\* 3.2.1
- gcc\* 4

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

### 1.4 Documentation

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

### 1.5 Technical Support

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

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

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

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

## 2 Installation

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

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

If you received the compiler product on DVD insert the DVD. Locate the disk image file (`m_cprof_p_11.1.xxx.dmg`) on the DVD and double-click. If you received the compiler product as a download, double-click the downloaded file, which will have a name of the form `m_cprof_p_11.1.xxx.dmg`.

Follow the prompts to complete installation.

## 2.1 Installation Folders

The 11.1 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.

- `<root>/intel/Compiler/11.1/xxx/`
  - `bin`
    - `ia32`
    - `intel64`
  - `include`
    - `ia32`
    - `intel64`
  - `lib`
  - `Frameworks`
    - `mkl`
  - `Documentation`
  - `man`
  - `Samples`

Where `<root>` is `/opt` by default, `xxx` is the three-digit build number and the folders under `bin`, `include` and `lib` are used as follows:

- `ia32`: Compilers that build applications that run on a 32-bit Intel-based Mac OS\* X system
- `intel64`: Compilers that build applications that run on a 64-bit Intel-based Mac OS\* X system (also referred to as Intel® 64 architecture)

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

## 2.2 Relocating Product After Install

If you wish to move the installed product's command line interface to a different location on disk, you can do so using a supplied script.

1. Open a terminal window
2. Change directory (`cd`) to the compiler install folder (for example, `/opt/intel/Compiler/11.1/xxx`)
3. Type the command:  
`./move_cprof.sh <new-install-location>`  
where `<new-install-location>` is the new directory path

This script will move all the files and update symlinks, environment variables and startup scripts as needed. If you have both Intel C++ and Intel Fortran installed in the old path, both products will be moved to the new location.

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

## 2.3 Removal/Uninstall

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

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

If you are not currently logged in as `root` you will be asked for the `root` password. 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 Mac OS\* X may be used in a build with version 11. Exceptions include:

- Objects built with the multi-file interprocedural optimization (`-ipo`) option must be recompiled.
- Objects built for 64-bit systems with the version 9.1 compiler and that use the `REAL(16)` or `REAL*16` datatypes must be recompiled.
- Modules that specified an `ATTRIBUTES ALIGN` directive and were compiled with versions earlier than 11 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 11.1 compiler, must be recompiled with version 11.1 Update 1 or later.

### 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. 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]old_logical_ldio`
- `-assume [no]old_maxminloc`
- `-fpe-all`
- `-ieee_fpe_flags`
- `-mkl[=lib]`

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.

### 3.4 Other Changes

#### 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 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.

#### 3.4.3 Environment Setup Script Changed

The `ifortvars.sh` (`ifortvars.csh`) script, used to set up the command-line build environment, changed in version 11.0. 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 /opt/intel/Compiler/11.1/xxx/bin/ifortvars.sh argument
```

Where *xxx* is the package identifier and *argument* is either `ia32` or `intel64` as described above under [Installation Folders](#). If you have installed the compiler into a different path, make the appropriate adjustments in the command. Establishing the compiler environment also establishes the Intel® Debugger (idb) environment.

## 3.5 Known Issues

### 3.5.1 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 `τ()` 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)

### 4.1 Known Problems

#### 4.1.1 Dwarf vs. Stabs Debug Formats

The debugger only supports debugging of executables whose debug information is in Dwarf format, and does not support the Stabs debug format. Use the `-gdwarf-2` flag on the compile command to have gcc and g++ generate Dwarf output. The Intel compilers (icc and ifort) produce Dwarf debug format with the `-g` flag.

#### 4.1.2 Compilation Requirements

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

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

```
ifort -g -o hello.exe hello.f90
```

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

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

```
ifort -c -g -o hello.o hello.f90
```

```
ifort -g -o hello.exe hello.o
```

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

```
ifort -g -save-temps -o hello.exe hello.f90
```

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

#### 4.1.3 Non-local Binary and Source File Access

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

```
Internal error: cannot create absolute path for: /home/me/hello
```

```
You cannot debug "/home/me/hello" because its type is "unknown".
```

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

```
Source file not found or not readable, tried...
```

```
./hello.f90
```

```
/auto/mount/site/foo/usr1/user_me/f_code/hello.f90
```

(Cannot find source file hello.f90)

The file-path specified will be correct.

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

#### 4.1.4 Local variables may not be visible

The linker on Mac OS X 10.5.4 (and subsequent versions) does not always issue definitions of local variables into the debug information in the executable. We do not have a characterization of when this occurs. The end result is that the variable is not visible or is visible but incorrectly evaluated.

The instances we have seen have involved local arrays in Fortran programs which were allocated in the `.bss` segment by the compiler. A work-around is to change the source to make the variable be global rather than local. In Fortran this is most easily done by putting the variable into a module or common block. Intel and Apple are working together to resolve this issue.

#### 4.1.5 Printing Fortran REAL\*16 variables

The debugger does not print the correct value for Fortran REAL\*16 variables.

#### 4.1.6 Debugging applications that fork

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

#### 4.1.7 Debugging applications that exec

The `$catchexecs` control variable is not supported.

#### 4.1.8 Fortran alternate entry points

Formal parameters of alternate entry points are not visible from within the debugger if they are not also formal parameters of the main entry point.

#### 4.1.9 Snapshots

Snapshots are not yet supported as described in the manual.

#### 4.1.10 Debugging optimized code

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

#### 4.1.11 Watchpoints

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

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

#### 4.1.12 Fortran modules and commons

A globally defined Fortran module should be rescoped with a double percent (%%) when referred to. For example, to set a breakpoint in the subroutine bar contained in a globally defined module foo, do

```
(idb) stop in foo%%bar
```

Please refer to the following section in the manual for the rescoping syntax:

Looking Around the Code, the Data and Other Process Information >

Looking at the Data >

The print Command

If you try to access (print, etc.) a Fortran module or common using the name in the source code, the debugger may not be able to find it. As a workaround, the you can try prepending '\_' to the name. For example, in the source code, if you have a common called "com":

```
(idb) print _com
```

#### 4.1.13 Graphical User Interface (GUI)

This version of the debugger does not support the GUI

#### 4.1.14 MPP Debugging Restrictions

MPP debugging is not supported as described in the manual.

#### 4.1.15 Function Breakpoints

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

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

#### 4.1.16 Core File Debugging

Debugging core files is not yet supported.

#### 4.1.17 Universal Binary Support

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

#### 4.1.18 Debugger variable `$threadlevel`

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

#### 4.1.19 Open File Descriptors Limitation

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

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

```
ulimit -n 2000
```

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

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

#### 4.1.20 `$cdir`, `$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.

#### 4.1.21 `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.

#### 4.1.22 `$stepg0` Default Value Changed

The debugger has changed the default value of the debugger variable `$stepg0` from 1 to 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 have compatibility with previous debugger versions as follows:

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

## 5 Intel® Math Kernel Library

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

### 5.1 Changes in This Version

#### 5.1.1 New features

- LAPACK 3.2
  - 238 new LAPACK functions
  - Extra Precise Iterative Refinement
  - Non-Negative Diagonals from Householder QR factorization
  - High Performance QR and Householder Reflections on Low-Profile Matrices
  - New fast and accurate Jacobi SVD
  - Routines for Rectangular Full Packed format
  - Pivoted Cholesky
  - Mixed precision iterative refinement (Cholesky)
  - More robust DQDS algorithm
- Introduced implementation of the DZGEMM Extended BLAS function (as described at <http://www.netlib.org/blas/blast-forum/>). See the description of the \*gemm family of functions in the BLAS section of the reference manual.
- PARDISO now supports real and complex, single precision data

#### 5.1.2 Usability/Interface improvements

- Sparse matrix format conversion routines which convert between the following formats:
  - CSR (3-array variation) ↔ CSC (3-array variation)
  - CSR (3-array variation) ↔ diagonal format
  - CSR (3-array variation) ↔ skyline
- Fortran95 BLAS and LAPACK compiled module files (.mod) are now included
  - Modules are pre-built with the Intel Fortran Compiler and are located in the include directory (see Intel® MKL User's Guide for full path)
  - Source is still available for use with other compilers
  - Documentation for these interfaces can be found in the Intel® MKL User's Guide
- The FFTW3 interface is now integrated directly into the main libraries
  - Source code is still available to create wrappers for use with compilers not compatible with the default Intel® Fortran compiler convention for name decoration
  - See Appendix G of the Reference Manual for information
- DFTI\_DESCRIPTOR\_HANDLE now represents a true type name and can now be referenced as a type in user programs
- Added parameter to Jacobi matrix calculation routine in the optimization solver domain to allow access to user data (see the description of the djacobix function in the reference manual for more information)

- Added an interface mapping calls to single precision BLAS functions in Intel® MKL (functions with “s” or “c” initial letter) to 64-bit floating point precision functions has been added on 64-bit architectures (See “sp2dp” in the Intel® MKL User Guide for more information)
- Compatibility libraries (also known as “dummy” libraries) have been removed from this version of the library

### 5.1.3 Performance improvements

- Further threading in BLAS level 1 and 2 functions for Intel® 64 architecture
  - Level 1 functions (vector-vector): (C,Z,S,D)ROT, (C,Z,S,D)COPY, and (C,Z,S,D)SWAP
    - Increase in performance by up to 1.7-4.7 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor depending on data location in cache
    - Increase in performance by up to 14-130 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
  - Level 2 functions (matrix-vector): (C,Z,S,D)TRMV, (S,D)SYMV, (S,D)SYR, and (S,D)SYR2
    - Increase in performance by up to 1.9-2.9 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor, depending on data location in cache
    - Increase in performance by up to 16-40 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
- Introduced recursive algorithm in 32-bit sequential version of DSYRK for up to 20% performance improvement on Intel® Core™ i7 processors and Intel® Xeon® processors in 5300, 5400, and 7400 series.
- Improved LU factorization (DGETRF) by 25% over Intel MKL 10.1 Update 1 for large sizes on the Intel® Xeon® 7460 Processor; small sizes are also dramatically improved
- BLAS \*TBMV/\*TBSV functions now use level 1 BLAS functions to improve performance by up to 3% on Intel® Core™ i7 processors and up to 10% on Intel® Core™2 processor 5300 and 5400 series.
- Improved threading algorithms to increase DGEMM performance
  - up to 7% improvement on 8 threads and up to 50% on 3,5,7 threads on the Intel® Core™ i7 processor
  - up to 50% improvement on 3 threads on Intel® Xeon® processor 7400 series.
- Threaded 1D complex-to-complex FFTs for non-prime sizes
- New algorithms for 3D complex-to-complex transforms deliver better performance for small sizes (up to 64x64x64) on 1 or 2 threads
- Implemented high-level parallelization of out-of-core (OOC) PARDISO when operating on symmetric positive definite matrices.
- Reduced memory use by PARDISO for both in-core and out-of-core on all matrix types

- PARDISO OOC now uses less than half the memory previously used in Intel MKL 10.1 for real symmetric, complex Hermitian, or complex symmetric matrices
- Parallelized Reordering and Symbolic factorization stage in PARDISO/DSS
- Up to 2 times better performance (30% improvement on average) on Intel® Core® i7 and Intel® Core™2 processors for the following VML functions:  $v(s,d)Round$ ,  $v(s,d)Inv$ ,  $v(s,d)Div$ ,  $v(s,d)Sqrt$ ,  $v(s,d)Exp$ ,  $v(s,d)Ln$ ,  $v(s,d)Atan$ ,  $v(s,d)Atan2$
- Optimized versions of the following functions available for Intel® Advanced Vector Extensions (Intel® AVX)
  - BLAS: DGEMM
  - FFTs
  - VML: exp, log, and pow
  - See important information in the Intel® MKL User's Guide regarding the `mkl_enable_instructions()` function for access to these functions

## 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](http://www.intel.com/software/products/mkl)) in both the product documentation and website.

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

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

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

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

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (<http://www.spiral.net/>) under license from Carnegie Mellon University. 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.

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.