

Intel® C++ Compiler 9.0 for Linux* Release Notes

Contents

- [Overview](#)
- [What's New](#)
- [System Requirements](#)
- [Installation](#)
- [Known Issues](#)
- [Technical Support](#)
- [Documentation](#)
- [Additional Information](#)
- [Disclaimer and Legal Information](#)

Overview

This product provides tools for Linux* software developers to create applications to run at top speeds on all Intel® IA-32 processors, Intel® processors with Intel® Extended Memory 64 Technology (Intel® EM64T) and the Intel® Itanium® processors. Optimizations include support for Intel® Streaming SIMD Extensions 2 (SSE2) in the Intel® Pentium® 4 and Pentium M processors, Intel® Streaming SIMD Extensions 3 (SSE3) in the Intel Pentium 4 processors with SSE3 support, and software pipelining in the Intel Itanium® 2 processor. Inter-procedural optimization (IPO) and profile-guided optimization (PGO) can provide greater application performance. Intel® Compilers support multi-threaded code development through autoparallelism and OpenMP* support.

The paper, *Optimizing Applications with the Intel® C++ and Fortran Compilers for Windows* and Linux**, explains how to use the Intel compilers to optimize for the Pentium 4 and Itanium processors and is available at <http://www.intel.com/software/products/compilers/>. Additional information on the Intel Software Development Products is available at <http://www.intel.com/software/products/>.

Product Contents

Intel® C++ Compiler for IA-32 Based Applications

The Intel® C++ Compiler for IA-32 based applications contains the following components:

- Intel® C++ Compiler for Linux for IA-32 applications, version 9.0
- Intel® Debugger for IA-32 applications, version 9.0
- Intel® Compiler code-coverage tool
- Intel® Compiler test-prioritization tool
- A version of the Eclipse* 3.0.1 Integrated Development Environment with C/C++ Development Tools 2.1.1 for the Intel C++ Compiler, and associated components
- The product documentation, version 9.0
 - The documentation index is provided for easy access of all the documents. It is located at `<install-dir>/doc/doc_index.htm`
 - A training tutorial *Enhancing Performance with Intel Compilers* is also included

Intel® C++ Compiler for Intel® EM64T-Based Applications

The Intel® C++ Compiler for Intel EM64T-based applications contains the following components:

- Intel® C++ Compiler for Linux for Intel EM64T-based applications, version 9.0
- Intel® Debugger for Intel EM64T-based applications, version 9.0
- Substitute headers for use with the Intel C++ Compiler, version 9.0
- Intel® Compiler code-coverage tool
- Intel® Compiler test-prioritization tool
- The product documentation, version 9.0
 - The documentation index is provided for easy access of all the documents. It is located at `<install-dir>/doc/doc_index.htm`
 - A training tutorial *Enhancing Performance with Intel Compilers* is also included.

Intel® C++ Compiler for Itanium®-Based Applications

The Intel® C++ Compiler for Itanium-based applications contains the following components:

- Intel® C++ Compiler for Linux for Itanium-based applications, version 9.0
- Intel® Debugger for Itanium-based applications, version 9.0
- Intel® Itanium Assembler to produce Itanium-based applications, version 8.0
- Substitute headers for use with the Intel C++ Compiler, version 9.0
- Intel® Compiler code-coverage tool
- Intel® Compiler test-prioritization tool
- The product documentation, version 9.0
 - The documentation index is provided for easy access of all the documents. It is located at `<install-dir>/doc/doc_index.htm`
 - A training tutorial *Enhancing Performance with Intel Compilers* is also included

Eclipse* Integrated Development Environment (IA-32 Only)

The Intel® C++ Compiler for Linux (IA-32 only) includes compiler integration with Eclipse* and the C/C++ Development Tools* (CDT). This functionality is an optional part of the compiler installation.

Eclipse is an open source software development project dedicated to providing a robust, full-featured, commercial-quality, industry platform for the development of highly integrated tools. It is an extensible, open source Integrated Development Environment (IDE).

The CDT (C/C++ Development Tools) project is dedicated to providing a fully functional C/C++ IDE for the Eclipse platform. CDT is layered on Eclipse, and provides a C/C++ development environment perspective.

The Intel C++ Compiler integration with the Eclipse/CDT IDE lets you develop, build, and run your Intel C/C++ projects in a visual, interactive environment.

See Also

- <http://www.eclipse.org/> for further information about Eclipse
- <http://www.eclipse.org/cdt/> for further information about CDT

What's New in Version 9.0

The following section discusses new features and changes in the Intel® C++ Compiler version 9.0 and updates to 9.0. Please see the separate release notes for the Intel® Debugger.

New Compiler Options

This section briefly lists compiler options that provide new functionality in this release.

Some compiler options are only available on certain systems, as indicated by these labels:

i32

The option is available on IA-32 based systems

i32em

The option is available on IA-32-based systems with Intel® Extended Memory 64 Technology (Intel® EM64T)

i64

The option is available on Itanium®-based systems

If no label appears, the option is available on all supported systems. If "only" appears in the label, the option is available on the identified system(s) only.

For more details on the options, refer to the Alphabetical Compiler Options section of the Intel C++ Compiler for Linux User's Guide.

-B<prefix>

Specifies where to find libraries, headers and executables for the compiler itself.

-debug [no]inline_debug_info

Specifies whether enhanced source position information is produced for inlined code.

-debug [no]semantic_stepping

Specifies whether enhanced debug information useful for breakpoints and stepping is produced.

-debug extended

Enables all -debug options except inline_debug_info.

-ffunction-sections

Places each function in its own comdat section.(C++ only)

-f[no-]math-errno

Tells the compiler that errno can be reliably tested after calls to standard math library functions.

-fno-builtin

Equivalent to -nolib_inline (a deprecated option).

-f[no-]exceptions

Enables or disables exception handling table generation.

-fno-gnu-keywords

Disables GNU keywords.

-fno-operator-names

Disables support for the operator names specified in the standard.

-f[no-]omit-frame-pointer (i32 only)

Indicates whether the compiler should use the frame pointer for IA32. -fno-omit-frame-pointer is a synonym for -fp.

-fpack-struct

Specifies that structure members should be packed together. It is the gcc spelling for -Zp1.

-fp-model double

Enables intermediates in 64-bit precision.

-fp-model [no-]except

Enables or disable floating point semantics.

-fp-model extended

Enables intermediates in 80-bit precision.

-fp-model fast

Allows value-unsafe optimizations.

-fp-model precise

Allows value-safe optimizations only.

-fp-model source
Enables intermediates in source precision.

-fp-model strict
Enables -fp-model precise, -fp-model except, as well as sensitivity to the FPU environment.
Disables floating point multiply add (FMA).

-ftemplate-depth-<val>
Controls the depth to which recursive templates are expanded.

-ftrapuv
Initializes stack local variables to an unusual value that may help detect uninitialized variables.

-funroll-loops
Tells the compiler to unroll user loops based on the default optimization heuristics. Synonym for -unroll.

-[no]gen-interfaces
Tells the compiler to generate an interface block for each routine in a source file.

-[no-]global-hoist
Determines whether certain optimizations are enabled that can move memory loads to a point earlier in the program execution than where they appear in the source.

-imacros
Allows a header to be specified that is included in front of the other headers in the translation unit; used in the Linux kernel build.

-iprefix <prefix>, -iwithprefix <dir>, -iwithprefixbefore <dir>
Options for indicating the prefix for referencing directories containing header files.

-i-dynamic
Links Intel-provided libraries dynamically (required if using -mcmmodel=medium or -mcmmodel=large for EM64T)

-i-static
Links Intel-provided libraries statically.

-malign-double (i32 only)
Aligns double, long double, and long long objects for better performance on IA32.

-map-opts
Enables a tool that maps a command line to another platform (Windows* to Linux* or vice versa).

-mfixed-range=f12-f15,f32-f127 (i64 only)
gcc spelling for the Itanium option -mP3OPT_ecg_reduced_fregs. Used in the Linux kernel build.

-m[no-]ieee-fp
Controls whether the compiler uses IEEE floating point comparisons.

-mtune=<keyword>
Performs optimizations for a particular CPU.

-[no-]multibyte-chars
Provides support for multi-byte characters.

-[no-]prec-sqrt (i32, i32em only)
Determines whether or not to disable certain optimizations that improve performance of square root operations while slightly reducing precision.

-print-multi-lib
Displays information about libraries being used.

-prof-gen-sampling
Asks the compiler to prepare the code for use with the sample-gathering tool profrun.

-sox
Causes the compiler to save compiler version and compilation options in the object and executable files (new for i64)

-ssp (i32 only)
Enables the software-based speculative pre-computation (SSP) optimization to generate prefetching helper threads.

--version
Displays compiler version information.

-W[no-]abi

- Determines whether tail-padding warnings are displayed.
- W[no-]deprecated
Enables or disables warnings related to deprecated features (places -U_DEPRECATED on the compiler command line).
 - Winline
Provides diagnostics about what is and is not inlined (depends on what IP functionality is available).
 - W[no-]missing-prototypes
Enables or disables warnings for missing prototypes.
 - W[no-]pointer-arith
Enables or disables warnings for questionable pointer arithmetic.
 - W[no-]uninitialized
Equivalent to -ww592 and -wd592.

Additional Optimization at -O2

When the default optimization level (-O2) is specified, the compiler now performs certain interprocedural and loop optimizations that were formerly performed only when -ip or -O3 were specified. We invite your comments as to the effect of this change on performance (both run-time and compile-time) as well as application results.

Software-based Speculative Pre-Computation (IA-32 Only)

The new -ssp option enables Software-based Speculative Pre-computation (SSP) optimization, which is also called Helper-Threading optimization. This feature provides a way to dynamically prefetch data cache blocks to reduce the impact of memory latency. It exploits the properties of source code constructs (such as delinquent loads and pointer-chasing loops) in applications.

SSP directly executes a subset of the original program instructions, called a slice, on separate threads alongside the main computation thread, in order to compute future memory accesses accurately. The helper threads run ahead of the main thread and trigger cache misses earlier on its behalf, thereby hiding the memory latency.

To be effective, SSP techniques require construction of efficient helper threads and processor-level support, such as Hyper-Threading Technology (HT Technology) support, which allows multiple threads to run concurrently with a shared cache. These techniques include:

- Delinquent load identification
- Loop selection
- Program slicing
- Helper-thread code generation

The results of SSP vary because each program has a different profile and different opportunities for SSP optimizations. For guidelines to help you determine if you can benefit by using SSP, see the topic "Software-based Speculative Precomputation (IA-32)" in the Profile-Guided Optimization section of your Optimizing Applications guide.

Loop Optimization Reports Improved

A number of improvements have been made to the optimization reports. For example, loop optimization reports, selected with -opt_report_phase hlo, now have more detailed line number information for unrolled loops, and identifiers for multiple versions of loops.

Multiple Object Default for `-ipo`

When `-ipo` is specified, the compiler now, by default, can produce multiple intermediate object files for input to the linker, rather than a single file. This can improve performance of the link phase. You can adjust this behavior by specifying a value for `-ipo`. Please refer to the Compiler Options Guide for more information.

Additional Warnings and Diagnostics

This version of the Intel C++ Compiler has improved diagnostic capabilities which may result in new informational, warning or error messages when your application is recompiled.

Itanium® Compatibility Issue Returning `struct` Containing `long double` Elements

On Itanium systems only, calls to a function whose return value was a `struct` that consisted solely of 80-bit floating point types (C/C++ `long double` or `long double complex`), and whose size was greater than 64 bytes but less than or equal to 256 bytes, incorrectly returned the function value on the stack instead of in floating point registers. For example:

```
typedef struct {  
    long double f1;  
    long double f2;  
    long double f3;  
    long double f4;  
    long double f5;  
} HFA;
```

```
HFA f(void);
```

The incorrect return value mechanism violates the Itanium calling conventions and is incompatible with `gcc` (which obeys the published calling conventions). The Intel C++ Compiler for Linux has been corrected in versions later than 8.1.023 so that these structs are returned properly, but this change makes such functions and calls to them incompatible with code compiled by version 8.1.023 and earlier. You should recompile all files that either call a function that returns such values, or contain such a function, using the latest version of the Intel C++ Compiler for Itanium-based Systems.

`GCC_EXEC_PREFIX` Environment Variable

If you specify a value for the `GCC_EXEC_PREFIX` environment variable, the compiler uses the value as a prefix to the following:

- `gcc` tools, such as `ld` and `as`
- Default include paths
- Default library paths

If the value specified is a directory, the compiler treats it as a path. For example, if `GCC_EXEC_PREFIX=/my/path/` then `gcc tool ld` becomes `/my/path/ld` and the compiler treats it as a path to the tool. Note that the tool must exist in the specified path or the path is ignored and the compiler searches for the tool in default directories.

Library search paths are modified to include the specified prefix for `crtd*.o` objects. The prefix is also used as an `-L` search path.

This feature is useful if you want to use an alternate set of gcc tools and libraries.

Note that if `-Ldir` or `-Bdir` is specified on the command line, they take precedence for library search paths over the environment variable.

The order of precedence is as follows (highest to lowest):

- `-Ldir`
- `-Bdir`
- `GCC_EXEC_PREFIX` environment variable

Small, Medium and Large Memory Models on EM64T

Applications built to take advantage of Intel EM64T can be built with one of three memory models:

Small (default)

Code and data is restricted to the first 2GB of address space, so that all accesses of code and data can be done with Instruction Pointer (IP)-relative addressing

Medium (`-mmodel=medium`)

Code is restricted to the first 2GB, no restriction on data; code can be accessed with IP-relative addressing, but access of data must use absolute addressing

Large (`-mmodel=large`)

No restrictions on code or data; accesses to both code and data use absolute addressing

IP-relative addressing requires only 32 bits, whereas absolute addressing requires 64-bits. This can affect code size and performance (IP-relative addressing is somewhat faster.)

Note: When the medium or large memory models are specified, you must also specify `-i-dynamic` to ensure that the correct dynamic versions of the Intel run-time libraries are used.

When shared objects (`.so`) are built, Position-Independent Code (PIC) is specified (`-fpic` is added by the compiler driver) so that a single `.so` can support all three memory models. However, code that is to be placed in a static library, or linked statically, must be built with the proper memory model specified. Note that there is a performance impact to specifying the Medium or Large memory models.

System Requirements

Supported Host and Target Combinations

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

IA-32 Host

Supported targets: IA-32 and Intel® EM64T

Intel® EM64T Host

Supported targets: IA-32 and Intel® EM64T

Intel® Itanium®-based System

Supported target: Intel® Itanium®-based System

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

Requirements to Develop IA-32 applications

- A system based on an IA-32 processor (minimum 450 MHz Intel Pentium® II processor or greater - Intel® Pentium® 4 or Pentium® D or Intel® Xeon™ processor recommended), or a system based on an Intel processor with Intel EM64T, or a system based on an AMD* Athlon* or AMD Opteron* processor
- 128 MB (256MB recommended).
- 100 MB of disk space, plus an additional 200 MB during installation for the download and temporary files.
- Linux system with glibc 2.2.4, 2.2.5, 2.2.93, 2.3.2 or 2.3.3 and the 2.4.X or 2.6.X Linux kernel as represented by the following distributions. **Note:** Not all distributions listed are validated and not all distributions are listed.
 - Red Hat* Linux 7.3, 8, 9
 - Red Hat Enterprise Linux* 2.1, 3, 4
 - SUSE* LINUX 8.2, 9.1
 - SUSE LINUX Enterprise Server* 8, 9
- Linux Developer tools component installed, including gcc, g++ and related tools.

Requirements to Develop Applications for Systems with Intel® EM64T or AMD Opteron Processors

- A system based on an Intel® processor with Intel EM64T or based on an AMD Opteron processor
- 256 MB of RAM (512 MB recommended)
- 300 MB free hard disk space, plus an additional 300 MB during installation for download and temporary files.
- 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
- Linux system with glibc 2.3.2 or 2.3.3 and the 2.4.X or 2.6.X Linux kernel as represented by the following Linux distributions, running in 64-bit mode. **Note:** Not all distributions listed are validated and not all distributions are listed.
 - Red Hat* Enterprise Linux 3, 4
 - SUSE* LINUX 9.1 Professional
 - SUSE LINUX Enterprise Server 9
- Linux Developer tools component installed, including gcc 3.3.3, g++ and related tools.
- 32-bit (IA-32) C and C++ runtime libraries: libm.so.6, libpthread.so.0, libc.so.6, libstdc++.so.5 and libgcc_s.so.1

Note: The requirement for the 32-bit (IA-32) libraries is due to the compiler and other tools being 32-bit applications that dynamically link to these libraries. If these libraries are not installed, the following error may be displayed when the compiler is invoked:

```
error while loading shared libraries: libstdc++.so.5: cannot open
shared object file: No such file or directory
```

The error message is confusing as it does not indicate that the IA-32 version of libstdc++.so.5 is required. To avoid this problem, be sure that the 32-bit (IA-32) versions of these libraries are installed. Most, but not all, Linux distributions for Intel EM64T will install these by default.

Requirements to Develop Itanium®-based Applications

- A system based on an Intel® Itanium® 2 processor.
- 512 MB (1GB recommended).
- 150 MB of disk space, plus an additional 200 MB during installation for the download and temporary files.
- Linux system with glibc 2.2.4, 2.2.5, 2.3.2 or 2.3.3 and the 2.4.X or 2.6.X Linux kernel as represented by the following distributions. **Note:** Not all distributions listed are validated and not all distributions are listed.
 - Red Hat Linux 7.2
 - Red Hat Enterprise Linux AS 2.1, 3, 4
 - SUSE LINUX Professional* 9.1
 - SUSE LINUX Enterprise Server 8, 9
 - United Linux* 1.0
- Linux Developer tools component installed, including gcc, g++ and related tools.

We recommend using binutils 2.14 or later, especially if using shared libraries as there are known issues with binutils 2.11.

We are unable to install on the unsupported configuration of Red Hat Enterprise Linux 4 beta systems without 32-bit support in the kernel. If you encounter this limitation please contact Intel® Premier Support for a workaround.

Notes:

- 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 contact Intel® Premier 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.

Installation

Please see the separate [Installation Guide](#) for information on installing the compiler and setting up the compiler environment. The default installation directories, referred to elsewhere in this document as `<install-dir>` and `<idb-install-dir>`, are:

- `/opt/intel/cc/9.0` (for IA-32 and Itanium)
- `/opt/intel/cce/9.0` (for Intel EM64T)
- `/opt/intel/idb/9.0` (for IA-32 and Itanium)
- `/opt/intel/idbe/9.0` (for Intel EM64T)

Known Issues

Header Incompatibility with RedHat* Enterprise Linux* 3, Update 4 (also SGI* ProPack* 3 Service Pack 5)

In Update 4 to RedHat* Enterprise Linux 3, inline assembly code was added to the file `/usr/include/c++/3.2.3/ia64_redhat-linux/bits/os_defines.h`. This causes the Intel® C++ Compiler to fail to compile sources referencing this header. Note that this problem is not

known to exist for any other version of Linux, including earlier versions of EL3 or beta versions of the next major release of RedHat Enterprise Linux. This issue also affects SGI ProPack 3 Service Pack 5.

A modified header file which corrects this problem is available from http://www.intel.com/software/products/compilers/downloads/os_defines.h.90

A good place to put the modified file is in the substitute headers directory of your installed compiler. For example, `<install-dir>/substitute_headers/c++/bits/os_defines.h`. The path must end with `bits/os_defines.h`. If you place it there the compiler will find it automatically. You should find an existing installer-created directory `<install-dir>/substitute_headers` and should create the sub-directory path `.../c++/bits` underneath it.

No support for Dinkumware* libraries for Intel® EM64T

The Dinkumware* libraries, which were supported on IA-32 and Itanium-based systems for earlier versions of the Intel C++ Compiler, are not supported for Intel EM64T.

-ipo_obj option is no longer supported

The `-ipo_obj` option, which forced generation of direct object code, is no longer supported. If the option is specified, a warning is given and the effect is as if `-ip` was specified instead.

OpenMP* limitations

POSIX threaded programs that require a large stack size may not run correctly on some versions of Linux because of hard-coded stack size limits in some versions of the Linux POSIX threads libraries. These limits also apply to OpenMP programs (`-openmp`) and automatically generated parallel programs (`-parallel`) with the Intel compilers, because the Intel compilers use the POSIX threads library to implement OpenMP based and automatically generated parallelism. Threaded programs that exceed the stack space limit usually experience segmentation violations or addressing errors.

To avoid these limitations, use a version of glibc built with the `FLOATING_STACKS` parameter defined. For some distributions, this implies using the shared rather than the static version of the pthreads library. Then use the `ulimit -s` or `limit stacksize` command to set the maximum shell stack size to an explicit large value, in units of KBytes, (not unlimited), and also set the `KMP_STACKSIZE` environment variable to the needed thread stacksize in bytes. Note, in the bash shell, `ulimit -s` can be used to set a large maximum stack size only once. In the C shell (csh), `limit stacksize`, with no dash before the argument, can be used to reset the maximum stacksize repeatedly.

This solution has been tested on glibc version 2.2.4-13 for IA-32 and glibc 2.2.4-19 for the Itanium Processor Family as found in the Red Hat 7.2 Linux distribution. For glibc 2.2.4-13 on IA-32, the shared version of the POSIX threads library must be used, (there should not be a `-static` flag in the compiler .cfg file or on the command line).

Compile time slow down when using both -g and inlining

There will be an increase in compile time when `-g` is used together with inlining. Inlining can happen if the user specifies `-ipo`, `-ip` or compiles a C++/C99 program at option levels `-O1` or above. This is due to the generation of debug information. For many applications, this combination of compiler options will not increase compile time or compile-time memory use.

Compiler hang on version query on glibc 2.2.4-26

We have identified a problem with glibc version 2.2.4-26 that shipped with the original version of Red Hat AS2.1. This version causes a compiler hang on the command "icc -v or icc -V (with no files to compile). Upgrading to glibc 2.2.4-31.7 fixes the problem. If you have taken any updates to your AS2.1 you will not see this problem. There was also a respin of the original AS2.1 that fixed this problem so only if you have a very early installation of AS2.1 that has never been updated will you see this issue.

`-relax` no longer passed to linker on Itanium®-based systems

As of version 9.0, the compiler driver no longer passes the `-relax` switch to the linker on Itanium-based systems, as this conflicts with the `-r` option. The `-relax` option is not needed as it is the default when using binutils 2.11.90.0.27 or later - 2.14 is recommended. If you must use an older binutils and wish to specify the `-relax` option, use `-Xlinker -relax` on the compile command which invokes the linker.

Limited debug information with automatic CPU dispatching (`-ax*`)

Compilation using `-ax{W|N|B|P}` results in two copies of generated code for each function. One for IA-32 generic code and one for CPU specific code. The symbol for each function then refers to an Auto CPU Dispatch routine that decides at run-time which one of the generated code sections to execute. Debugger breakpoints that are set on these functions by name cause the application to stop in the dispatch routine. This may cause unexpected behavior when debugging. This issue may be addressed in a future version of the Intel Debugger and Compilers.

Cannot debug or view traceback for IA-32 programs built without `-fip`

Compilation using `-fip` specifies that the IA-32 EBP register be used as a frame pointer rather than a general purpose register. Debuggers and traceback handlers may not be able to properly unwind through a stack that contains a call to a function that is compiled without `-fip` in effect. If you compile with `-g` or `-O0`, `-fip` is implicitly enabled, but not if you specify a higher optimization level explicitly (such as `-O2`). If you intend to use the debugger or traceback on an application, and are using some level of optimization higher than `-O0`, you should also specify `-fip` to ensure that the debugger and traceback handler can use frame pointers.

GNU assembler may not recognize `-xP` generated code

Older versions of the GNU Assembler may not be able to process assembly code generated by compiling with the `-[a]xP` option. Use binutils version 2.14.90.0.4.1 or later, or FSFbinutils 2.15 or later if this is an issue for you.

Using older `gdb` versions with Intel® Compilers

Intel compilers for Linux generate Dwarf2-format debugging information, including several advanced features in Dwarf2 such as declarations nested within classes. Older `gdb` debuggers, such as version 5.3.90-*, are sometimes unable to correctly handle these Dwarf features. For best success on source code which uses the full expressiveness of the C++ language, please consider using `gdb` version 6.1 or newer.

Use `idb` with Extended Debug Information

If you use the `-debug` keywords `inline_debug_info`, `semantic_stepping`, `variable_locations` or `extended`, you should use the Intel Debugger (`idb`), as other debuggers may not understand the extended information and may behave unpredictably. We are working with the developers of other debuggers towards their adding support for the extended debug information.

Technical Support

Your feedback is very important to us. To receive technical support for the tools provided in this product and technical information including FAQ's and product updates, you need to be registered for an Intel® Premier Support account on our secure web site, <https://premier.intel.com>. Please register at <https://registrationcenter.intel.com/>.

- Registering for support varies for release products or pre-release products (alpha, beta, etc) - only released products have support web pages on <http://support.intel.com/>.
- If you are having trouble registering or are unable to access your Intel® Premier Support account, please let Intel know of the problem at <https://registrationcenter.intel.com/support>.

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

For information about the Intel® C++ Compiler's Users Forums, FAQ's, tips and tricks, and other support information, please visit: <http://support.intel.com/support/performance/c/linux/>. For general support information please visit <http://www.intel.com/software/products/support/>.

Submitting Issues

Steps to submit an issue:

1. Go to <https://premier.intel.com/>.
2. Log in to the site. Note that your username and password are case-sensitive.
3. Click on the "Go" button next to the "Product" drop-down list.
4. Click on the "Submit Issue" link in the left navigation bar.
5. Choose "Development Environment (tools, SDV, EAP)" from the "Product Type" drop-down list.
6. If this is a software or license-related issue, choose "Intel(R) C++ Compiler, Linux*" from the "Product Name" drop-down list.
7. Enter your question and complete the fields in the windows that follow to successfully submit the issue.

Guidelines for problem report or product suggestion:

1. Describe your difficulty or suggestion.
For problem reports please be as specific as possible, so that we may reproduce the problem. For compiler problem reports, please include the compiler options and a small test case if possible.
2. Describe your system configuration information.
Get the version of `glibc` and kernel with following commands:

```
> uname -a  
> rpm -qa | grep glibc
```


If you don't have `rpm` installed, use the command below:

```
> ls /lib/libc*
```

and copy the information into the corresponding Intel Premier Support fields.

Get the Intel C++ Compiler's Package ID with the following commands:

```
> icc -V
```

and copy the "Package ID" (e.g. `l_cc_p_9.0.xxx`) from the output into the corresponding Intel Premier Support field. Please include any other specific information that may be relevant to helping us to reproduce and address your concern.

3. If you were not able to install the compiler or cannot get the Package ID, enter the filename you downloaded as the package ID.

Resolved Issues

Please review `<package ID>_README` (e.g. `l_cc_p_9.0.xxx_README`), available for download from Intel® Premier Support, <https://premier.intel.com/>, to see which issues have been resolved in the latest version of the compiler.

Documentation

You can view the Intel compiler and related HTML-based documentation with your Web browser. You should use a Web browser that supports JavaScript (such as Firefox*), so it can which provide full navigation, search, index look-up, and hyperlink capabilities amongst the online help files PDF versions of most manuals are available online at

<http://developer.intel.com/software/products/compilers/clin/docs/manuals.htm> .

The documentation is installed in the `<install-dir>/doc` directory. An HTML index document can be found at `<install-dir>/doc/doc_index.htm` . A training tutorial *Enhancing Performance with Intel® Compilers* is also available from links in the documentation index. *The Intel® Debugger Manual* is provided in HTML form in the Intel® Debugger doc directory.

For information on the GNU glibc C language library, documentation can be obtained from the Linux OS vendor or from the GNU web site, www.gnu.org.

Viewing Manpages

The `icc(1)` manpage provides a list of command-line options and related information for the `icc` and `icpc` compiler commands. To display the `icc(1)` manpage, type the following command after you set up your environment by using a source command to execute the `<install-dir>/bin/iccvars.*sh` file:

```
$ man icc
```

The `man` command provides single keys or key combinations that let you scroll through the displayed content, search for a string, jump to a location, and perform other functions. For example, type the `z` to view the next screen or `w` to view the previous screen. To obtain help about the `man` command, type the `h` key; when you are done viewing help, type the `q` key to return to the displayed manpage. To search, type `/ character` followed by the search string (`/string`) and press Enter. After viewing the man command text, type `q` to return to the shell command prompt.

Viewing Documentation

The HTML documentation format has been tested to work with web browsers shipped on supported Linux* distributions. PDF versions of the compiler documentation are available at: <http://developer.intel.com/software/products/compilers/clin/docs/manuals.htm>

Additional Information

Related Products and Services

Information on Intel® software development products is available at <http://www.intel.com/software/products>.

Some of the related products include:

- The [Intel® Software College](#) provides training for developers on leading-edge software development technologies. Training consists of online and instructor-led courses covering all Intel architectures, platforms, tools, and technologies.
- The [Intel® VTune™ Performance Analyzer](#) enables you to evaluate how your application is utilizing the CPU and helps you determine if there are modifications you can make to improve your application's performance.
- The [Intel® C++ and Fortran Compilers](#) are an important part of making software run at top speeds with full support for the latest Intel IA-32 and Itanium® processors.
- The [Intel® Performance Library Suite](#) provides a set of routines optimized for various Intel processors. The [Intel® Math Kernel Library](#), which provides developers of scientific and engineering software with a set of linear algebra, fast Fourier transforms and vector math functions optimized for the latest Intel Pentium® and Intel Itanium processors. The [Intel® Integrated Performance Primitives](#) consists of cross-platform tools to build high performance software for several Intel architectures and several operating systems.

Disclaimer and Legal Information

The information in this document is subject to change without notice and Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document. This document and the software described in it are furnished under license and may only be used or copied in accordance with the terms of the license. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. The information in this document is provided in connection with Intel products and should not be construed as a commitment by Intel Corporation.

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. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

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 software described in this document may contain software defects which may cause the product to deviate from published specifications. Current characterized software defects are available on request.

Intel, the Intel logo, Intel SpeedStep, Intel NetBurst, Intel NetStructure, MMX, Intel386, Intel486, Celeron, Intel Centrino, Intel Xeon, Intel XScale, Itanium, Pentium, Pentium II Xeon, Pentium III Xeon, Pentium M, and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Copyright © 2005, Intel Corporation.