



Migrating Applications from Intel® Fortran Compiler 7.1 to Intel Fortran Compiler 9.0

Contents

Executive Summary	3
Introduction	3
Source Changes.....	3
Representation of 'Logical' Values	3
How Many Bytes in That 'RECL='?.....	4
'GETARG,' 'IARGC' and 'NARGS' Are Now Intrinsic.....	4
Padding Changes in 'Common'	4
Build Changes	4
Compile Command Is Now 'ifort'	4
Building Mixed Language Programs.....	4
Switches Not Implemented.....	5
Product Features	5
Getting Help	5
Conclusion	5

Executive Summary

Intel® Fortran Compiler 9.0 delivers extremely high performance and integrates seamlessly with other developer tools. This new version provides full support for cutting-edge hardware technologies such as Intel® multi-core processors and Intel processors supporting Intel® EM64T, as well as continuing support for previous 32-bit and 64-bit platforms. It also brings many new features and expanded industry compatibility, along with improved performance.

This new compiler version builds on the significant advances made in version 8.0, which combined previous Intel and Compaq technologies. Migrating applications from Intel Fortran compiler version 7.1 to later versions may require changes in sources and build methods. This white paper details those changes to accommodate version 9.0.

Introduction

The current versions of Intel Fortran Compiler for Linux* and Intel® Visual Fortran for Windows* merge the best Fortran technologies of Intel and Compaq (now part of Hewlett-Packard). Intel code generation, optimization, and parallel processing technologies are now combined with the features, extensions, and language processing know-how of Compaq Visual Fortran to create a series of compatible Fortran compilers across IA-32 and Itanium®-based systems — both for Windows and Linux — that offer an industry-leading feature set, along with outstanding runtime performance.

While in most cases, existing applications can simply be rebuilt with the new compiler without source changes, some applications may need minor coding changes, and build methods may need minor adjustments. This paper describes key differences you are likely to encounter. For additional details, please refer to the appropriate compiler release notes and the compiler documentation. These documents are installed along with the compiler and are also available on the Intel Web site at <http://www.intel.com/software/products/>.

This paper applies to those coming to the version 9.0 compiler from version 7.1 of the Intel Fortran Compiler for either Windows or Linux. If you are porting from Compaq Visual Fortran, please refer to the separate document *Porting Applications from Compaq Visual Fortran to Intel Visual Fortran Compilers*. Where appropriate, items that apply to only one of the platforms are so noted.

Source Changes

Intel has taken great care to avoid the need for source changes - a key design goal for each new version of the compilers is to permit the capability to “rebuild and go.” Due to a number of implementation differences between the Intel and Compaq compilers, however, changes may be needed for some applications when migrating from older versions of the Intel Fortran Compiler.

An important point to keep in mind is that the 8.0 (or later) compiler detects many more incorrect language usages than did previous versions of the Intel Fortran Compiler. You may get error messages, either at compile time or at link time, which you did not see before. If you believe that any of these behaviors are incorrect, please contact Intel® Premier Support at <http://premier.intel.com/>.

Representation of ‘Logical’ Values

The Fortran standard does not specify the internal binary representation of LOGICAL values and does not allow conversion between LOGICAL and numeric data types. However, some Fortran programmers have incorporated into their applications non-standard assumptions about the internal representation of LOGICAL values. Most often when mixing Fortran and C, compiler writers have obliged by permitting free conversion between LOGICAL and numeric types.

In previous versions of the Intel Fortran Compilers, a value tested as false if it contained a binary zero, and tested as true if it was non-zero; this convention matches the standard definition of the Boolean type in C. Similarly, the constant `.FALSE.` was zero and the constant `.TRUE.` was 1, if converted to an integer. Version 8.0 and later compilers adopt the Compaq Visual Fortran model, where even integer values (low bit clear) are false, odd integer values (low bit set) are true, and `.TRUE.` is -1. (`.FALSE.` is zero in both models.)

If your application performs logical tests on numeric values, such as Boolean values returned from a call to a C routine, you should rewrite the code to compare against an appropriate numeric constant. As an alternative, you can instruct the compiler to adopt the zero/non-zero model by specifying the compiler option `/fpscomp:logicals` (Windows) or `-fpscomp:logicals` (Linux).

How Many Bytes in That ‘RECL=’?

When opening a file for unformatted access and specifying the record length using the `RECL=` keyword, the Fortran standard does not specify the units of the value supplied, other than as “processor-dependent units.” Previous versions of the Intel Fortran Compiler interpreted this value as being the number of bytes, while Compaq Visual Fortran used four-byte “numeric storage units” instead. (Earlier versions of the Fortran standard said “storage units.”) The 8.0 and later Intel Compilers adopt the Compaq interpretation, which means that if your program opens an unformatted file using `RECL=`, typically for direct access, the record may be four times longer than desired.

To accommodate this change, either divide the `RECL=` value by four, or specify one of the following options:

- `-assume byterecl` (Linux)
- `/assume:byterecl` (Windows).

‘GETARG,’ ‘IARGC’ and ‘NARGS’ Are Now Intrinsic

The compiler now recognizes the command-line inquiry routines `GETARG`, `IARGC`, and `NARGS` as intrinsic procedures. If your application source declares any of these names as `EXTERNAL` or provides an explicit procedure interface for them, you must remove those declarations to prevent link-time errors.

Padding Changes in ‘Common’

Previous versions of the Intel Fortran Compiler inserted up to seven bytes of padding between variables in `COMMON` to ensure that 8-byte numeric variables were aligned on natural boundaries. This practice violated the Fortran standard’s rules on common association. The default behavior in version 9.0 is not to insert

padding between variables in a `COMMON` block. The compiler issues warning messages (unless suppressed by `-warn noalign` or `/warn:noalign`) if it sees an unaligned numeric variable. This change should be invisible to most applications, except for those that access variables in a `COMMON` block other than by their names. This would be the case, for example, when the `COMMON` block is shared with a C extern.

If you want to pad variables in `COMMON`, specify `-align commons` (`/align:commons`) to align on up to 4-byte boundaries, or `-align dcommons` (`/align:dcommons`) to align on up to 8-byte boundaries (same as in version 7.1).

Build Changes

In most cases, you will not need to make changes to your build procedures. To provide for compatibility with future versions of Intel Fortran compilers, however, some changes are recommended. This section describes differences that affect building applications.

Compile Command Is Now ‘ifort’

Version 7.1 of the Intel Fortran Compiler used various names to invoke the compiler from the command line: `ifl` and `efl` on Windows, and `ifc` and `efc` on Linux. In version 9.0, the compile command is now `ifort` across all platforms. This should make it easier to develop applications on multiple platforms. The appropriate compiler is selected by using the specific “vars” script, such as `ifortvars.bat` (Windows) or `ifortvars.sh` (Linux), when beginning your command session.

In version 8.0 and later, the old names are still accepted, but they will generate an informational message alerting you to the new name. You can suppress this message with `/quiet` (Windows) or `-quiet` (Linux).

Building Mixed Language Programs

If you are building a mixed-language program on Linux where the main entry point is not in Fortran, add the `-nofor_main` switch to the `ifort` (or `ifc`) command, which builds the executable. This avoids an error message from ID regarding “doubly defined symbol main”. This item is not applicable on Windows.

Switches Not Implemented

The following switches are not currently implemented in the Intel Fortran compilers:

- `-CA`, `-CS`, `-CV` (runtime correctness checks; `-CB` and `-CU` are supported)

Product Features

Intel Fortran Compiler for Linux and Intel Visual Fortran for Windows provide a rich feature set that delivers winning performance for both legacy and cutting-edge technologies:

- Full support of Intel multi-core processors and Intel processors supporting Intel EM64T, along with previous Intel processors and architectures.
- Complete source compatibility with Compaq Visual Fortran.
- Support for the Fortran 2003 feature “Enhanced Data Type Facilities” (TR 15581). This allows you to have `ALLOCATABLE` arrays as dummy arguments, function results, and components of a derived type.
- The Traceback feature, which, upon encountering a runtime error, displays the source-file name and line number at the point of error, as well as those of the calling routines.

Getting Help

If you need assistance, help is available. For “how to” questions, visit the Intel Fortran Compiler user forums at <http://softwareforums.intel.com/>. There you can ask questions of other knowledgeable users and search for previous discussions that may address your problem. If you believe you have found a bug in the product,

Intel Premier Support is ready to assist. You must first register for support; you are prompted to do so when you install the compiler, or you can go to the Registration Center at <http://www.intel.com/software/products/registrationcenter/index.htm>. Once registered, log in at <https://premier.intel.com/> and submit your support request.

Conclusion

The Intel Fortran Compiler 9.0 provides clear advantages, enhancing application performance and ensuring support for emerging Intel® hardware platforms such as Intel multi-core processors and Intel processors supporting Intel EM64T. The new release also continues Intel’s commitment to providing better levels of optimization and performance, as well as compatibility with previous generations of hardware.

By following the best practices introduced in this guide, developers can smooth their migration to this latest compiler and derive its advantages efficiently. The following additional resources will help you take the next steps toward adopting these tools today:

- Intel® Software Development Products: <http://www.intel.com/software/products/compilers/>
- Intel® Software Tools Developer Center: <http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/tools/index.htm>



Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95052-8119
USA

For product and purchase information visit:
www.intel.com/software/products

Intel, the Intel logo, Pentium, and Itanium 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.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL 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. Intel products are not intended for use in medical, life saving, or life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.