

Intel® Compiler Code Coverage Tool

Table of Contents

Overview	3
Features.....	3
Default Colors for the Code Coverage Tool	4
Compatibility	5
World-Class Technical Support	5

Overview

When it is time to test software quality, developers can use the Code-Coverage Tool, which is included in all Intel® compilers, to easily see which parts of an application have been tested and which haven't. The tool paints a picture of code-use by displaying color-coded, annotated HTML pages that provide summary information and simplified navigation through most- and least-covered modules and functions.

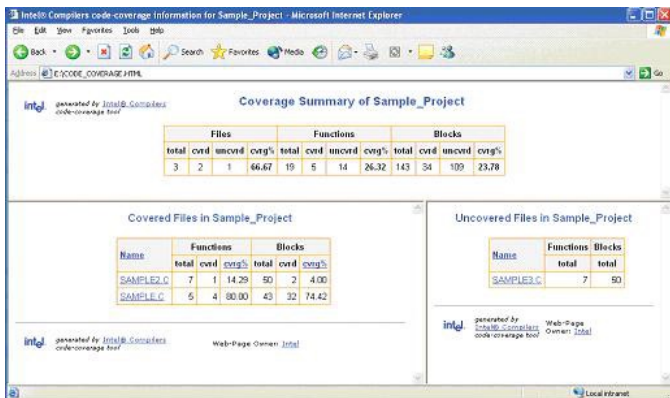
Features

The Code Coverage Tool can improve development efficiency, reduce defects, and improve application performance. Here's how:

Flexible Analysis: The Code Coverage Tool allows developers to analyze selected application modules or entire applications. If a subset of application modules is analyzed, the tool generates coverage information only for those modules, avoiding the overhead that would otherwise be incurred in analyzing the full application. Alternatively, developers can analyze the entire application, a subset, or break the covered modules into different components and use the Code Coverage Tool to obtain coverage information about each individual module.

Improved Testing

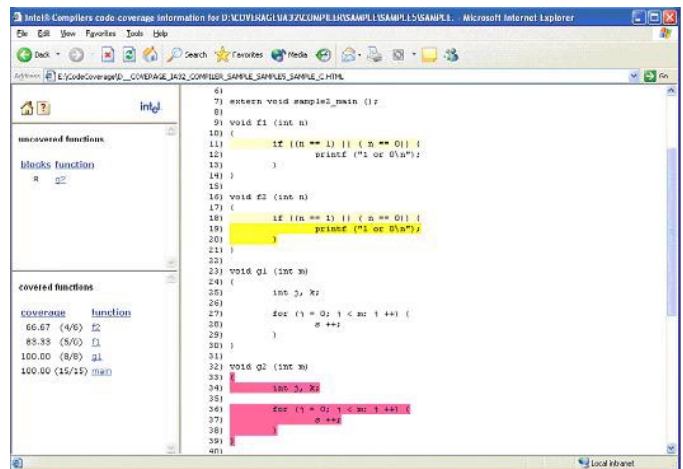
- **Easy-to-Use 'Top-Level' Report:** The top-level summary offers an overall code coverage report for the modules specified in the analysis. The summary information includes the number of modules exercised by your tests and, just as important, those not exercised. It also includes information about the total number of functions and blocks in each module and the portions that were covered



Top-level Coverage Summary of a Sample Project.

The screen above shows an example of a code coverage summary for a sample project. The workload applied in this test exercised 34 of 143 blocks, representing 5 of 19 functions in two of three modules. In the file, SAMPLE.C, four of five functions were exercised.

- **Helper Frames:** The Code Coverage Tool creates frames that help developers locate uncovered code. The top frame displays the list of uncovered functions while the bottom frame displays the list of covered functions.
 - For uncovered functions, the tool displays the number of basic blocks for each function.
 - For covered functions, the tool displays the number of blocks, the number of covered blocks, and their ratio (the coverage rate).
 - Sort coverage lists. You can sort these lists based on the coverage rate, number of blocks, or function names. With just one click, you can see the least-covered function in the list, and with another click, it displays the source code of the function, allowing you to browse through the function body.
- **Source Code Reports:** Clicking on a module name in the top-level summary (for example, SAMPLE.C in the image above), produces a browser-based view of the module source code that is highlighted to show the code that was exercised by the tests. The color coding also tells you the extent to which that code was tested.



Source View of a Sample Module.

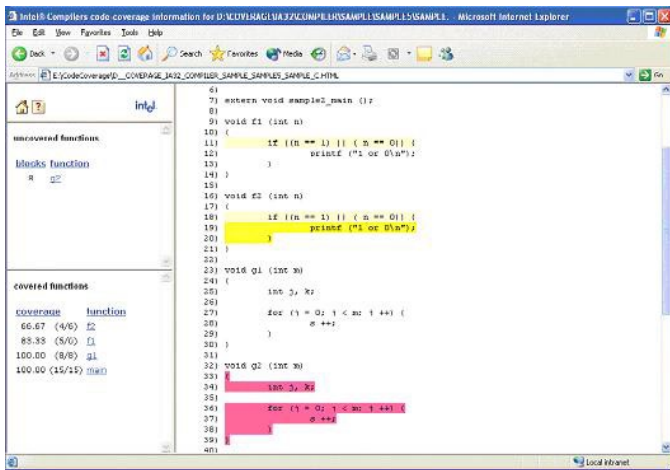
Clicking on a module name (SAMPLE.C) in the top-level coverage summary produces a listing that highlights the exercised code. In this example, the beige-highlighted code was partially covered, the yellow was run but not exercised by any of the tests the developer set up, and the pink was never exercised.

- **Coloring Scheme for Code Coverage Tool:** The Code Coverage Tool uses color to distinguish between covered code, uncovered basic blocks, uncovered functions, and partially covered code. Developers can change colors to suit tastes or as may be required for review of printed reports.

Default Colors for the Code Coverage Tool

Default Color	Meaning
Covered code	The portion of code highlighted in this color was exercised by the tests. To change this color, use the -ccolor switch.
Uncovered basic block	Basic blocks highlighted in this color were not exercised by any of the tests. They were exercised, however, within functions that were executed during the tests. To change this color, use the -bcolor switch.
Uncovered function	Functions highlighted in this color were never called during the tests. To change this color, use the -fcolor switch.
Partially-covered code	More than one basic block was generated for the code highlighted in this color. Some of the blocks were covered while some were not. To change this color, use the -pcolor switch.
Unknown	No code was generated for the source highlighted in this color. Most likely, this source is a comment, a header-file inclusion, or a variable declaration. To change this color, use the -ucolor switch.

- Dynamic Counters:** Developers can configure the Code Coverage Tool to display dynamic execution counts. This information appears directly beneath the source position where the corresponding basic block begins. If more than one basic block is generated for the code at a source position, then the number of generated blocks and the number of the executed blocks are displayed as well.



Dynamic-execution information is displayed in lines 11 and 12 and shows that the if-statement on line 11 was executed 10 times, that two basic blocks were generated for it but only one of the two blocks was executed, which causes it to be color-coded to reflect partial coverage. The information in line 12 indicates that variable n had a value of 0 or 1 only seven of the ten times the if-statement was executed.

- Differential Coverage:** The Code Coverage Tool can be used to compare the test-profiles of a primary and secondary run. This feature helps developers find portions of code not exercised by tests in the primary run that are exercised in the second run. For example, developers may want to compare coverage provided by a standard test-suite and a workload provided by a customer or compare coverage as changes are made to internal test-suites.
- Included with Intel® Compilers:** The Code Coverage and Test Prioritization Tools seamlessly support C, C++ and Fortran languages and are included with and integrated into Intel compiler products using IA-32 or Itanium® processor families running either Windows or Linux operating environments. When you buy an Intel compiler for these systems, you automatically get the Code Coverage Tool and the Test Prioritization Tool - there is nothing more to buy! Both are seamlessly integrated into and support Intel® C, C++, and Fortran compilers
- Support:** Every purchase of Intel® Software Development Products includes one year of support services, which provides access to Intel® Premier Support and all product updates during that time. Because the Code Coverage and Test Prioritization Tools are included in Intel Compilers, customers should purchase the compiler and register for support. This will automatically register you for Code Coverage and Test Prioritization Tools support. Customers get one Premier Support account for each license they purchase. You can renew at the end of the year at a reduced rate. For details on the support services, see the support section of this Web site.

Please visit the Registration Center at <https://registrationcenter.intel.com/RegCenter/Register.aspx> to initiate product support or click here for further information.

Compatibility

Intel Compilers' Code Coverage and Test Prioritization Tools complement the set of tools that help developers easily create the fastest software on Intel® architecture. They can dramatically reduce testing turn-around time and are included with all Intel Compiler products. The compiler products provide an optimizing compiler that increases the performance of applications on Intel processor-based systems. The compilers are compatible with leading development environments and support the features of Intel's latest processors. All compilers come with one year of Intel Premier Support, including new versions and updates.

World-Class Technical Support

With the purchase of Intel® Software Development Products, you will receive one year of technical support and product updates from Intel® Premier Support, our interactive issue management and communication web site. This premium support service allows you to submit questions, download product updates, and access technical notes, application notes, and other documentation. For more information, visit the Intel Registration Center at <https://registrationcenter.intel.com/RegCenter/Register.aspx>

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel® Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

