**GCC 6 Release Series**

**Changes, New Features, and Fixes**

This page is a brief summary of some of the huge number of improvements in GCC 6. For more information, see the [Porting to GCC 6](https://gcc.gnu.org/gcc-6/porting_to.html) page and the [full GCC documentation](https://gcc.gnu.org/onlinedocs/index.html#current).

..................................................................

**Caveats**

* The default mode for C++ is now -std=gnu++14 instead of -std=gnu++98.
* Support for a number of older systems and recently unmaintained or untested target ports of GCC has been declared obsolete in GCC 6. Unless there is activity to revive them, the next release of GCC will have their sources permanently **removed**. The following ports for individual systems on particular architectures have been obsoleted:
  + SH5 / SH64 (sh64-\*-\*) as announced [here](https://gcc.gnu.org/ml/gcc/2015-08/msg00101.html).
* The AVR port requires binutils version 2.26.1 or later for the fix for [PR71151](https://gcc.gnu.org/bugzilla/show_bug.cgi?id=71151) to work.

..................................................................

**General Optimizer Improvements**

* UndefinedBehaviorSanitizer gained a new sanitization option, -fsanitize=bounds-strict, which enables strict checking of array bounds. In particular, it enables -fsanitize=bounds as well as instrumentation of flexible array member-like arrays.
* Type-based alias analysis now disambiguates accesses to different pointers. This improves precision of the alias oracle by about 20-30% on higher-level C++ programs. Programs doing invalid type punning of pointer types may now need -fno-strict-aliasing to work correctly.
* Alias analysis now correctly supports weakref and alias attributes. This makes it possible to access both a variable and its alias in one translation unit which is common with link-time optimization.
* Value range propagation now assumes that the this pointer of C++ member functions is non-null. This eliminates common null pointer checks but also breaks some non-conforming code-bases (such as Qt-5, Chromium, KDevelop). As a temporary work-around -fno-delete-null-pointer-checks can be used. Wrong code can be identified by using -fsanitize=undefined.
* Link-time optimization improvements:
  + warning and error attributes are now correctly preserved by declaration linking and thus -D\_FORTIFY\_SOURCE=2 is now supported with -flto.
  + Type merging was fixed to handle C and Fortran interoperability rules as defined by the Fortran 2008 language standard. As an exception, CHARACTER(KIND=C\_CHAR) is not inter-operable with char in all cases because it is an array while char is scalar. INTEGER(KIND=C\_SIGNED\_CHAR) should be used instead. In general, this inter-operability cannot be implemented, for example, on targets where function passing conventions of arrays differs from scalars.
  + More type information is now preserved at link time reducing the loss of accuracy of the type based alias analysis compared to builds without link-time optimization.
  + Invalid type punning on global variables and declarations is now reported with -Wodr-type-mismatch.
  + The size of LTO object files was reduced by about 11% (measured by compiling Firefox 46.0).
  + Link-time parallelization (enabled using -flto=n) was significantly improved by decreasing the size of streamed data when partitioning programs. The size of streamed IL while compiling Firefox 46.0 was reduced by 66%.
  + The linker plugin was extended to pass information about type of binary produced to GCC back end (that can be also manually controlled by -flinker-output). This makes it possible to properly configure the code generator and support incremental linking. Incremental linking of LTO objects by gcc -r is now supported on plugin-enabled setups. There are two ways to perform incremental linking:
    1. Linking by ld -r will result in an object file with all sections from individual object files mechanically merged. This delays the actual link time optimization to final linking step and thus permits whole program optimization. Linking final binary with such object files is however slower.
    2. Linking by gcc -r will lead to link time optimization and produce final binary into the object file. Linking such object file is fast but avoids any benefits from whole program optimization.
  + GCC 7 will support incremental link-time optimization with gcc -r.
* Inter-procedural optimization improvements:
  + Basic jump threading is now performed before profile construction and inline analysis, resulting in more realistic size and time estimates that drive the heuristics of the of inliner and function cloning passes.
  + Function cloning now more aggressively eliminates unused function parameters.

..................................................................

**New Languages and Language specific improvements**

<ul> <li> Compared to GCC 5, the GCC 6 release series includes a much improved implementation of the [OpenACC 2.0a specification](http://www.openacc.org/). Highlights are:

* In addition to single-threaded host-fallback execution, offloading is supported for nvptx (Nvidia GPUs) on x86\_64 and PowerPC 64-bit little-endian GNU/Linux host systems. For nvptx offloading, with the OpenACC parallel construct, the execution model allows for an arbitrary number of gangs, up to 32 workers, and 32 vectors.
* Initial support for parallelized execution of OpenACC kernels constructs:
  + Parallelization of a kernels region is switched on by -fopenacc combined with -O2 or higher.
  + Code is offloaded onto multiple gangs, but executes with just one worker, and a vector length of 1.
  + Directives inside a kernels region are not supported.
  + Loops with reductions can be parallelized.
  + Only kernels regions with one loop nest are parallelized.
  + Only the outer-most loop of a loop nest can be parallelized.
  + Loop nests containing sibling loops are not parallelized.
* Typically, using the OpenACC parallel construct gives much better performance, compared to the initial support of the OpenACC kernels construct.
* The device\_type clause is not supported. The bind and nohost clauses are not supported. The host\_data directive is not supported in Fortran.
* Nested parallelism (cf. CUDA dynamic parallelism) is not supported.
* Usage of OpenACC constructs inside multithreaded contexts (such as created by OpenMP, or pthread programming) is not supported.
* If a call to the acc\_on\_device function has a compile-time constant argument, the function call evaluates to a compile-time constant value only for C and C++ but not for Fortran.

See the [OpenACC](https://gcc.gnu.org/wiki/OpenACC) and [Offloading](https://gcc.gnu.org/wiki/Offloading) wiki pages for further information. </li> </ul> <h3 id="ada">Ada</h3>

**C family**

* Version 4.5 of the [OpenMP specification](http://openmp.org/wp/openmp-specifications/) is now supported in the C and C++ compilers.
* The C and C++ compilers now support attributes on enumerators. For instance, it is now possible to mark enumerators as deprecated: enum {
* newval,
* oldval \_\_attribute\_\_ ((deprecated ("too old")))
* };
* Source locations for the C and C++ compilers are now tracked as ranges, rather than just points, making it easier to identify the subexpression of interest within a complicated expression. For example: **test.cc:** In function **'int test(int, int, foo, int, int)'**:
* **test.cc:5:16:** **error:** no match for **'operator\*'** (operand types are **'int'** and **'foo'**)
* return p + **q \* r** \* s + t;
* **~~^~~**
* In addition, there is now initial support for precise diagnostic locations within strings: **format-strings.c:3:14:** **warning:** field width specifier **'\*'** expects a matching **'int'** argument [**-Wformat=**]
* printf("%\*d");
* **^**
* Diagnostics can now contain "fix-it hints", which are displayed in context underneath the relevant source code. For example: this is currently the only example in the tree; various others are pending **fixits.c:** In function **'bad\_deref'**:
* **fixits.c:11:13:** **error:** **'ptr'** is a pointer; did you mean to use **'->'**?
* return ptr**.**x;
* **^**
* **->**
* The C and C++ compilers now offer suggestions for misspelled field names: **spellcheck-fields.cc:52:13:** **error:** **'struct s'** has no member named **'colour'**; did you mean **'color'**?
* return ptr->**colour**;
* **^~~~~~**
* also, pending patch to add fix-it hints to the above
* New command-line options have been added for the C and C++ compilers:
  + -Wshift-negative-value warns about left shifting a negative value.
  + -Wshift-overflow warns about left shift overflows. This warning is enabled by default. -Wshift-overflow=2 also warns about left-shifting 1 into the sign bit.
  + -Wtautological-compare warns if a self-comparison always evaluates to true or false. This warning is enabled by -Wall.
  + -Wnull-dereference warns if the compiler detects paths that trigger erroneous or undefined behavior due to dereferencing a null pointer. This option is only active when -fdelete-null-pointer-checks is active, which is enabled by optimizations in most targets. The precision of the warnings depends on the optimization options used.
  + -Wduplicated-cond warns about duplicated conditions in an if-else-if chain.
  + -Wmisleading-indentation warns about places where the indentation of the code gives a misleading idea of the block structure of the code to a human reader. For example, given [CVE-2014-1266](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1266): **sslKeyExchange.c:** In function **'SSLVerifySignedServerKeyExchange'**:
  + **sslKeyExchange.c:629:3:** **warning:** this **'if'** clause does not guard... [**-Wmisleading-indentation**]
  + **if** ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
  + **^~**
  + **sslKeyExchange.c:631:5:** **note:** ...this statement, but the latter is misleadingly indented as if it is guarded by the **'if'**
  + **goto** fail;
  + **^~~~**
  + This warning is enabled by -Wall.
* The C and C++ compilers now emit saner error messages if merge-conflict markers are present in a source file. **test.c:3:1:** **error:** version control conflict marker in file
* **<<<<<<<** HEAD
* **^~~~~~~**

**C**

* It is possible to disable warnings when an initialized field of a structure or a union with side effects is being overridden when using designated initializers via a new warning option -Woverride-init-side-effects.
* A new type attribute scalar\_storage\_order applying to structures and unions has been introduced. It specifies the storage order (aka endianness) in memory of scalar fields in structures or unions.

**C++**

* The default mode has been changed to -std=gnu++14.
* [C++ Concepts](http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4377.pdf) are now supported when compiling with -fconcepts.
* -flifetime-dse is more aggressive in dead-store elimination in situations where a memory store to a location precedes a constructor to that memory location.
* G++ now supports [C++17](https://gcc.gnu.org/projects/cxx-status.html#cxx1z.html) fold expressions, u8 character literals, extended static\_assert, and nested namespace definitions.
* G++ now allows constant evaluation for all non-type template arguments.
* G++ now supports C++ Transactional Memory when compiling with -fgnu-tm.

**Runtime Library (libstdc++)**

* Extensions to the C++ Library to support mathematical special functions (ISO/IEC 29124:2010), thanks to Edward Smith-Rowland.
* Experimental support for C++17, including the following new features:
  + std::uncaught\_exceptions function (this is also available for -std=gnu++NN modes);
  + new member functions try\_emplace and insert\_or\_assign for unique\_key maps;
  + non-member functions std::size, std::empty, and std::data for accessing containers and arrays;
  + std::invoke;
  + std::shared\_mutex;
  + std::void\_t and std::bool\_constant metaprogramming utilities.
* Thanks to Ville Voutilainen for contributing many of the C++17 features.
* An experimental implementation of the File System TS.
* Experimental support for most features of the second version of the Library Fundamentals TS. This includes polymorphic memory resources and array support in shared\_ptr, thanks to Fan You.
* Some assertions checked by Debug Mode can now also be enabled by \_GLIBCXX\_ASSERTIONS. The subset of checks enabled by the new macro have less run-time overhead than the full \_GLIBCXX\_DEBUG checks and don't affect the library ABI, so can be enabled per-translation unit.
* Timed mutex types are supported on more targets, including Darwin.
* Improved std::locale support for DragonFly and FreeBSD, thanks to John Marino and Andreas Tobler.

**Fortran**

* Fortran 2008 SUBMODULE support.
* Fortran 2015 EVENT\_TYPE, EVENT\_POST, EVENT\_WAIT, and EVENT\_QUERY support.
* Improved support for Fortran 2003 deferred-length character variables.
* Improved support for OpenMP and OpenACC.
* The MATMUL intrinsic is now inlined for straightforward cases if front-end optimization is active. The maximum size for inlining can be set to n with the -finline-matmul-limit=n option and turned off with -finline-matmul-limit=0.
* The -Wconversion-extra option will warn about REAL constants which have excess precision for their kind.
* The -Winteger-division option has been added, which warns about divisions of integer constants which are truncated. This option is included in -Wall by default.

<h3 id="go">Go</h3> <h3 id="java">Java (GCJ)</h3> ..................................................................

**libgccjit**

* The driver code is now run in-process within libgccjit, providing a small speed-up of the compilation process.
* The API has gained entrypoints for
  + [timing how long was spent in different parts of code](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/jit/topics/performance.html),
  + [creating switch statements](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/jit/topics/functions.html#gcc_jit_block_end_with_switch),
  + [allowing unreachable basic blocks in a function](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/jit/topics/contexts.html#gcc_jit_context_set_bool_allow_unreachable_blocks), and
  + [adding arbitrary command-line options to a compilation](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/jit/topics/contexts.html#gcc_jit_context_add_command_line_option).

..................................................................

**New Targets and Target Specific Improvements**

**AArch64**

* A number of AArch64-specific options have been added. The most important ones are summarised in this section but for usage instructions please refer to the documentation.
* The new command-line options -march=native, -mcpu=native and -mtune=native are now available on native AArch64 GNU/Linux systems. Specifying these options causes GCC to auto-detect the host CPU and choose the optimal setting for that system.
* -fpic is now supported when generating code for the small code model (-mcmodel=small). The size of the global offset table (GOT) is limited to 28KiB under the LP64 SysV ABI, and 15KiB under the ILP32 SysV ABI.
* The AArch64 port now supports target attributes and pragmas. Please refer to the [documentation](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/gcc/AArch64-Function-Attributes.html#AArch64-Function-Attributes) for details of available attributes and pragmas as well as usage instructions.
* Link-time optimization across translation units with different target-specific options is now supported.
* The option -mtls-size= is now supported. It can be used to specify the bit size of TLS offsets, allowing GCC to generate better TLS instruction sequences.
* The option -fno-plt is now fully functional.
* The ARMv8.1-A architecture and the Large System Extensions are now supported. They can be used by specifying the -march=armv8.1-a option. Additionally, the +lse option extension can be used in a similar fashion to other option extensions. The Large System Extensions introduce new instructions that are used in the implementation of atomic operations.
* The ACLE half-precision floating-point type \_\_fp16 is now supported in the C and C++ languages.
* The ARM Cortex-A35 processor is now supported via the -mcpu=cortex-a35 and -mtune=cortex-a35 options as well as the equivalent target attributes and pragmas.
* The Qualcomm QDF24xx processor is now supported via the -mcpu=qdf24xx and -mtune=qdf24xx options as well as the equivalent target attributes and pragmas.
* Code generation for the ARM Cortex-A57 processor is improved. Among general code generation improvements, a better algorithm is added for allocating registers to floating-point multiply-accumulate instructions offering increased performance when compiling with -mcpu=cortex-a57 or -mtune=cortex-a57.
* Code generation for the ARM Cortex-A53 processor is improved. A more accurate instruction scheduling model for the processor is now used, and a number of compiler tuning parameters have been set to offer increased performance when compiling with -mcpu=cortex-a53 or -mtune=cortex-a53.
* Code generation for the Samsung Exynos M1 processor is improved. A more accurate instruction scheduling model for the processor is now used, and a number of compiler tuning parameters have been set to offer increased performance when compiling with -mcpu=exynos-m1 or -mtune=exynos-m1.
* Improvements in the generation of conditional branches and literal pools were made to allow the compiler to compile functions of a large size. Constant pools are now placed into separate rodata sections. The new option -mpc-relative-literal-loads is introduced to generate per-function literal pools, limiting the maximum size of functions to 1MiB.
* Several correctness issues with generation of Advanced SIMD instructions for big-endian targets have been fixed resulting in improved code generation for ACLE intrinsics with -mbig-endian.

**ARM**

* Support for revisions of the ARM architecture prior to ARMv4t has been deprecated and will be removed in a future GCC release. The -mcpu and -mtune values that are deprecated are: arm2, arm250, arm3, arm6, arm60, arm600, arm610, arm620, arm7, arm7d, arm7di, arm70, arm700, arm700i, arm710, arm720, arm710c, arm7100, arm7500, arm7500fe, arm7m, arm7dm, arm7dmi, arm8, arm810, strongarm, strongarm110, strongarm1100, strongarm1110, fa526, fa626. The value arm7tdmi is still supported. The values of -march that are deprecated are: armv2,armv2a,armv3,armv3m,armv4.
* The ARM port now supports target attributes and pragmas. Please refer to the [documentation](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/gcc/ARM-Function-Attributes.html#ARM-Function-Attributes) for details of available attributes and pragmas as well as usage instructions.
* Support has been added for the following processors (GCC identifiers in parentheses): ARM Cortex-A32 (cortex-a32), ARM Cortex-A35 (cortex-a35). The GCC identifiers can be used as arguments to the -mcpu or -mtune options, for example: -mcpu=cortex-a32 or -mtune=cortex-a35.

<h3 id="avr">AVR</h3>

**Heterogeneous Systems Architecture**

* GCC can now generate HSAIL (Heterogeneous System Architecture Intermediate Language) for simple OpenMP device constructs if configured with --enable-offload-targets=hsa. A new libgomp plugin then runs the HSA GPU kernels implementing these constructs on HSA capable GPUs via a standard HSA run time. If the HSA compilation back end determines it cannot output HSAIL for a particular input, it gives a warning by default. These warnings can be suppressed with -Wno-hsa. To give a few examples, the HSA back end does not implement compilation of code using function pointers, automatic allocation of variable sized arrays, functions with variadic arguments as well as a number of other less common programming constructs. When compilation for HSA is enabled, the compiler attempts to compile composite OpenMP constructs #pragma omp target teams distribute parallel for  into parallel HSA GPU kernels.

**IA-32/x86-64**

* GCC now supports the Intel CPU named Skylake with AVX-512 extensions through -march=skylake-avx512. The switch enables the following ISA extensions: AVX-512F, AVX512VL, AVX-512CD, AVX-512BW, AVX-512DQ.
* Support for new AMD instructions monitorx and mwaitx has been added. This includes new intrinsic and built-in support. It is enabled through option -mmwaitx. The instructions monitorx and mwaitx implement the same functionality as the old monitor and mwait instructions. In addition mwaitx adds a configurable timer. The timer value is received as third argument and stored in register %ebx.
* x86-64 targets now allow stack realignment from a word-aligned stack pointer using the command-line option -mstackrealign or \_\_attribute\_\_ ((force\_align\_arg\_pointer)). This allows functions compiled with a vector-aligned stack to be invoked from objects that keep only word-alignment.
* Support for address spaces \_\_seg\_fs, \_\_seg\_gs, and \_\_seg\_tls. These can be used to access data via the %fs and %gs segments without having to resort to inline assembly. Please refer to the [documentation](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/gcc/Named-Address-Spaces.html#Named-Address-Spaces) for usage instructions.
* Support for AMD Zen (family 17h) processors is now available through the -march=znver1 and -mtune=znver1 options.

<h3 id="mips">MIPS</h3>

**MeP**

* Support for the MeP (mep-elf) architecture has been deprecated and will be removed in a future GCC release.

**MSP430**

* The MSP430 compiler now has the ability to automatically distribute code and data between low memory (addresses below 64K) and high memory. This only applies to parts that actually have both memory regions and only if the linker script for the part has been specifically set up to support this feature. A new attribute of either can be applied to both functions and data, and this tells the compiler to place the object into low memory if there is room and into high memory otherwise. Two other new attributes - lower and upper - can be used to explicitly state that an object should be placed in the specified memory region. If there is not enough left in that region the compilation will fail. Two new command-line options - -mcode-region=[lower|upper|either] and -mdata-region=[lower|upper|either] - can be used to tell the compiler what to do with objects that do not have one of these new attributes.

<h3 id="nds32">NDS32</h3>

**PowerPC / PowerPC64 / RS6000**

* PowerPC64 now supports IEEE 128-bit floating-point using the \_\_float128 data type. In GCC 6, this is **not** enabled by default, but you can enable it with -mfloat128. The IEEE 128-bit floating-point support requires the use of the VSX instruction set. IEEE 128-bit floating-point values are passed and returned as a single vector value. The software emulator for IEEE 128-bit floating-point support is only built on PowerPC GNU/Linux systems where the default CPU is at least power7. On future ISA 3.0 systems (POWER 9 and later), you will be able to use the -mfloat128-hardware option to use the ISA 3.0 instructions that support IEEE 128-bit floating-point. An additional type (\_\_ibm128) has been added to refer to the IBM extended double type that normally implements long double. This will allow for a future transition to implementing long double with IEEE 128-bit floating-point.
* Basic support has been added for POWER9 hardware that will use the recently published OpenPOWER ISA 3.0 instructions. The following new switches are available:
  + -mcpu=power9: Implement all of the ISA 3.0 instructions supported by the compiler.
  + -mtune=power9: In the future, apply tuning for POWER9 systems. Currently, POWER8 tunings are used.
  + -mmodulo: Generate code using the ISA 3.0 integer instructions (modulus, count trailing zeros, array index support, integer multiply/add).
  + -mpower9-fusion: Generate code to suitably fuse instruction sequences for a POWER9 system.
  + -mpower9-dform: Generate code to use the new D-form (register+offset) memory instructions for the vector registers.
  + -mpower9-vector: Generate code using the new ISA 3.0 vector (VSX or Altivec) instructions.
  + -mpower9-minmax: Reserved for future development.
  + -mtoc-fusion: Keep TOC entries together to provide more fusion opportunities.
* New constraints have been added to support IEEE 128-bit floating-point and ISA 3.0 instructions:
  + wb: Altivec register if -mpower9-dform is enabled.
  + we: VSX register if -mpower9-vector is enabled for 64-bit code generation.
  + wo: VSX register if -mpower9-vector is enabled.
  + wp: Reserved for future use if long double is implemented with IEEE 128-bit floating-point instead of IBM extended double.
  + wq: VSX register if -mfloat128 is enabled.
  + wF: Memory operand suitable for POWER9 fusion load/store.
  + wG: Memory operand suitable for TOC fusion memory references.
  + wL: Integer constant identifying the element number mfvsrld accesses within a vector.
* Support has been added for \_\_builtin\_cpu\_is() and \_\_builtin\_cpu\_supports(), allowing for very fast access to AT\_PLATFORM, AT\_HWCAP, and AT\_HWCAP2 values. This requires use of glibc 2.23 or later.
* All hardware transactional memory builtins now correctly behave as memory barriers. Programmers can use #ifdef \_\_TM\_FENCE\_\_ to determine whether their "old" compiler treats the builtins as barriers.
* Split-stack support has been added for gccgo on PowerPC64 for both big- and little-endian (but **not** for 32-bit). The gold linker from at least binutils 2.25.1 must be available in the PATH when configuring and building gccgo to enable split stack. (The requirement for binutils 2.25.1 applies to PowerPC64 only.) The split-stack feature allows a small initial stack size to be allocated for each goroutine, which increases as needed.
* GCC on PowerPC now supports the standard lround function.
* A new configuration option ---with-advance-toolchain=at was added for PowerPC 64-bit GNU/Linux systems to use the header files, library files, and the dynamic linker from a specific Advance Toolchain release instead of the default versions that are provided by the GNU/Linux distribution. In general, this option is intended for the developers of GCC, and it is not intended for general use.
* The "q", "S", "T", and "t" asm-constraints have been removed.
* The "b", "B", "m", "M", and "W" format modifiers have been removed.

**S/390, System z, IBM z Systems**

* Support for the IBM z13 processor has been added. When using the -march=z13 option, the compiler will generate code making use of the new instructions and registers introduced with the vector extension facility. The -mtune=z13 option enables z13 specific instruction scheduling without making use of new instructions.
* Compiling code with -march=z13 reduces the default alignment of vector types bigger than 8 bytes to 8. This is an ABI change and care must be taken when linking modules compiled with different arch levels which interchange variables containing vector type values. For newly compiled code the GNU linker will emit a warning.
* The -mzvector option enables a C/C++ language extension. This extension provides a new keyword vector which can be used to define vector type variables. (Note: This is not available when enforcing strict standard compliance e.g. with -std=c99. Either enable GNU extensions with e.g. -std=gnu99 or use \_\_vector instead of vector.)
* Additionally a set of overloaded builtins is provided which is partially compatible to the PowerPC Altivec builtins. In order to make use of these builtins the vecintrin.h header file needs to be included.
* The new command line options -march=native, and -mtune=native are now available on native IBM z Systems. Specifying these options will cause GCC to auto-detect the host CPU and rewrite these options to the optimal setting for that system. If GCC is unable to detect the host CPU these options have no effect.
* The IBM z Systems port now supports target attributes and pragmas. Please refer to the [documentation](https://gcc.gnu.org/onlinedocs/gcc-6.1.0/gcc/S_002f390-Function-Attributes.html#S_002f390-Function-Attributes) for details of available attributes and pragmas as well as usage instructions.
* -fsplit-stack is now supported as part of the IBM z Systems port. This feature requires a recent gold linker to be used.
* Support for the g5 and g6 -march=/-mtune= CPU level switches has been deprecated and will be removed in a future GCC release. -m31 from now on defaults to -march=z900 if not specified otherwise. -march=native on a g5/g6 machine will default to -march=z900.

<h3 id="rx">RX</h3>

**SH**

* Support for SH5 / SH64 has been declared obsolete and will be removed in future releases.
* Support for the FDPIC ABI has been added. It can be enabled using the new -mfdpic target option and --enable-fdpic configure option.

**SPARC**

* An ABI bug has been fixed in 64-bit mode. Unfortunately, this change will break binary compatibility with earlier releases for code it affects, but this should be pretty rare in practice. The conditions are: a 16-byte structure containing a double or a 8-byte vector in the second half is passed to a subprogram in slot #15, for example as 16th parameter if the first 15 ones have at most 8 bytes. The double or vector was wrongly passed in floating-point register %d32 in lieu of on the stack as per the SPARC calling conventions.

..................................................................

**Operating Systems**

**AIX**

* DWARF debugging support for AIX 7.1 has been enabled as an optional debugging format. A more recent Technology Level (TL) and GCC built with that level are required for full exploitation of DWARF debugging capabilities.

<h3 id="dragonfly">DragonFly BSD</h3> <h3 id="freebsd">FreeBSD</h3>

**Linux**

* Support for the [musl C library](http://www.musl-libc.org/) was added for the AArch64, ARM, MicroBlaze, MIPS, MIPS64, PowerPC, PowerPC64, SH, i386, x32 and x86\_64 targets. It can be selected using the new -mmusl option in case musl is not the default libc. GCC defaults to musl libc if it is built with a target triplet matching the \*-linux-musl\* pattern.

**RTEMS**

* The RTEMS thread model implementation changed. Mutexes now use self-contained objects defined in Newlib <sys/lock.h> instead of Classic API semaphores. The keys for thread specific data and the once function are directly defined via <pthread.h>. Self-contained condition variables are provided via Newlib <sys/lock.h>. The RTEMS thread model also supports C++11 threads.
* OpenMP support now uses self-contained objects provided by Newlib <sys/lock.h> and offers a significantly better performance compared to the POSIX configuration of libgomp. It is possible to configure thread pools for each scheduler instance via the environment variable GOMP\_RTEMS\_THREAD\_POOLS.

**Solaris**

* Solaris 12 is now fully supported. Minimal support had already been present in GCC 5.3.
* Solaris 12 provides a full set of startup files (crt1.o, crti.o, crtn.o), which GCC now prefers over its own ones.
* Position independent executables (PIE) are now supported on Solaris 12.
* Constructor priority is now supported on Solaris 12 with the system linker.
* libvtv has been ported to Solaris 11 and up.

<h3 id="vxmils">VxWorks MILS</h3>

**Windows**

* The option -mstackrealign is now automatically activated in 32-bit mode whenever the use of SSE instructions is requested.

.................................................................. <h2>Documentation improvements</h2> ..................................................................

**Other significant improvements**

* The gcc and g++ driver programs will now provide suggestions for misspelled command line options. $ gcc -static-libfortran test.f95
* gcc: **error:** unrecognized command line option **'-static-libfortran'**; did you mean **'-static-libgfortran'**?
* The --enable-default-pie configure option enables generation of PIE by default.

**GCC 6.2**

This is the [list of problem reports (PRs)](https://gcc.gnu.org/bugzilla/buglist.cgi?bug_status=RESOLVED&resolution=FIXED&target_milestone=6.2) from GCC's bug tracking system that are known to be fixed in the 6.2 release. This list might not be complete (that is, it is possible that some PRs that have been fixed are not listed here).

**Target Specific Changes**

**SPARC**

* Support for --with-cpu-32 and --with-cpu-64 configure options has been added on bi-architecture platforms.
* Support for the SPARC M7 (Niagara 7) processor has been added.
* Support for the VIS 4.0 instruction set has been added.

====================================================================

*For questions related to the use of GCC, please consult these web pages and the* [*GCC manuals*](https://gcc.gnu.org/onlinedocs/)*. If that fails, the* [*gcc-help@gcc.gnu.org*](mailto:gcc-help@gcc.gnu.org) *mailing list might help. Comments on these web pages and the development of GCC are welcome on our developer list at* [*gcc@gcc.gnu.org*](mailto:gcc@gcc.gnu.org)*. All of* [*our lists*](https://gcc.gnu.org/lists.html) *have public archives.*

Copyright (C) [Free Software Foundation, Inc.](http://www.fsf.org/) Verbatim copying and distribution of this entire article is permitted in any medium, provided this notice is preserved.

These pages are [maintained by the GCC team](https://gcc.gnu.org/about.html). Last modified 2016-08-17 IGNORE DIFF [.](http://validator.w3.org/check/referer)