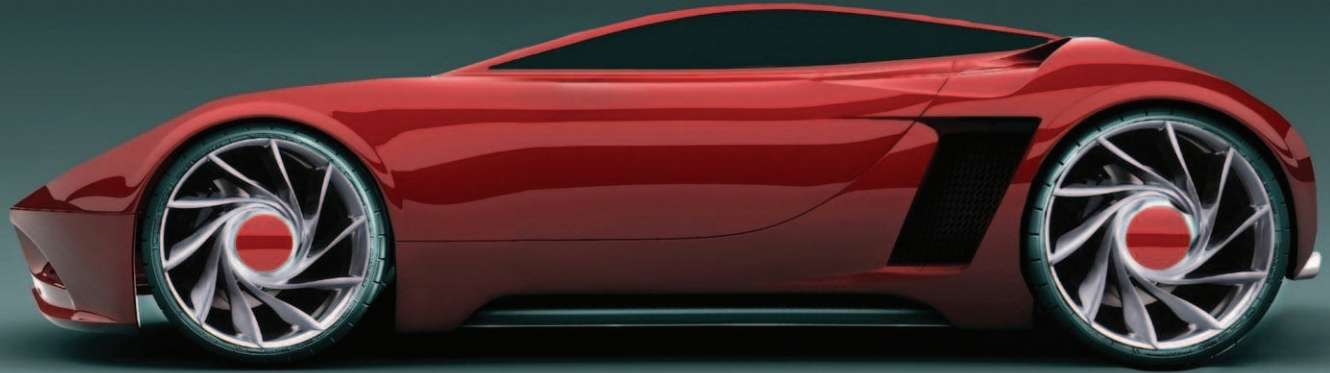


Compilador GNU 4.3

¡OPTIMIZADO!



STEFAN emmanuel, Fotolia

El último GCC 4.3 está destinado a hacerse con el mundo de la programación gracias a sus nuevas optimizaciones, al soporte experimental para la próxima generación de C++ 200x, una STL opcional de C++ concurrente, y un nuevo compilador de Java cortesía del proyecto Eclipse. **POR RENÉ REBE**

Poco después de la aparición de GCC 4.2, ya se encuentra disponible la versión 4.3 [2] del compilador de GNU [1]. Como cabía esperar, se han eliminado muchas de las funciones etiquetadas como obsoletas, como las opciones de optimización `-m386`, `-m486`, `-mpentium` y `-mpentiumpro`. Si realmente necesita las optimizaciones para estas CPUs, pueden reanimarse por medio de las opciones `-march=` y `-mtune=`. Los usuarios con CPUs nuevas apreciarán las opciones de optimización dedicadas a Geode de AMD y Core 2 de Intel, así como las características SSE3 (`-msse3`), SSE4.1 (`-msse4.1`) y SSE4.2 (`-msse4.2`).

Teniendo en cuenta la arquitectura, GCC 4.3 añade ARM versión 7 y la extensión Thumb-2 para la optimización del tamaño. GCC también incluye soporte directo para la Synergistic Processor Unit (SPU) Cell Broadband Engine Architecture de IBM, que se encuentra tanto en la Playstation 3 como en los servidores de IBM. Las notas de prensa [2] detallan varios cambios para MIPS, Motorola 68000, Coldfire, Cris y PowerPC.

Algunas de las nuevas optimizaciones se basan en la librería MPFR (Multiple-Precision Floating point computations with correct Rounding [3]), la cual ayuda a GCC a evaluar expresiones complejas, realizando llamadas a funciones matemá-

ticas incluidas y truncándolas a funciones equivalentes o constantes en tiempo de compilación. La librería MPFR devuelve resultados correctos, independientes de la precisión del punto flotante y de la CPU de destino. Como contrapunto, se crea una dependencia con las librerías MPFR, y por ello, con las librerías GMP, haciendo que la compilación cruzada de GCC sea más compleja, ya que estas dos librerías también necesitan compilarse con el código C++.

La Optimización x86 Revela Errores en el Kernel

El código GCC 4.3 para x86 ya no genera una instrucción explícita `clt` antes de cada operación de cadena de autorepetición (`REP MOV ...`), ahorrándose de este modo entre 4 y 52 ciclos en un Intel Pentium. Este desarrollo ha revelado que algunos kernels de Linux y BSD no reinician el flag de dirección por ellos mismos durante la gestión de señales. Esto significa que el kernel realiza operaciones de cadena en los manejadores de señales en la dirección contraria, lo que lleva a un direccionamiento erróneo – una vulnerabilidad [4] bastante obvia.

Nuevos Desarrollos en la Compilación de C

La versión 4.3 de GCC es la primera en detectar los accesos fuera de rango de los

arrays en tiempo de compilación con el uso de constantes o de desplazamientos. La opción para ello es `-Warray-bounds`, que también se activa con `-Wall`.

La nueva aritmética en punto flotante [5] añade más precisión para las aplicaciones financieras y científicas. El uso de la base 10 en vez de la base 2 también implica que las operaciones son más precisas a la hora de aplicar el redondeo; por ejemplo, el resultado de 0.9:10 es ahora 0.09 y no 0.089999996.

Una extensión de GCC permite a los desarrolladores especificar las constantes enteras binarias con el prefijo `0b` o mapas de bits con `0B`. El soporte de tipo de datos en punto fijo de la especificación Embedded C se encuentra disponible, aunque sólo ha sido implementada para MIPS.

C++ Conforme a la ISO

C++ implementa ahora más componentes de la próxima generación del estándar ISO 200x, o C++ 0x para abreviar. Para habilitar el estándar [6] están disponibles las opciones `-std=c++0x` o `-std=gnu++0x`. C++ soporta en estos momentos plantillas con un número variable de parámetros y “assertions” estáticas. En caso de tener plantillas anidadas, los programadores ya no tendrán que insertar espacios en blanco entre los corchetes angulares:

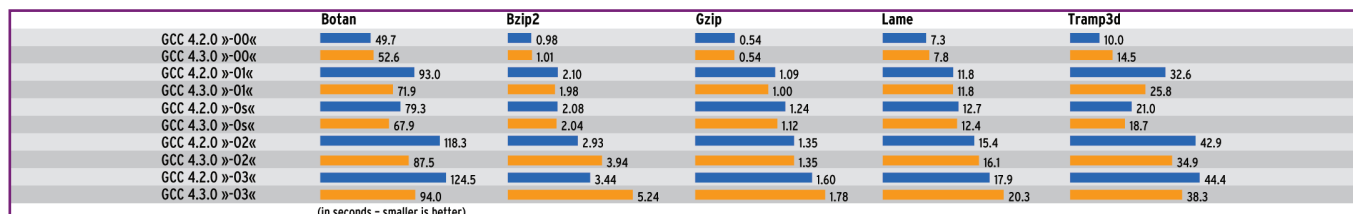


Figura 1: Comparativa de los tiempos de compilación entre GCC 4.2 y la última versión 4.3.

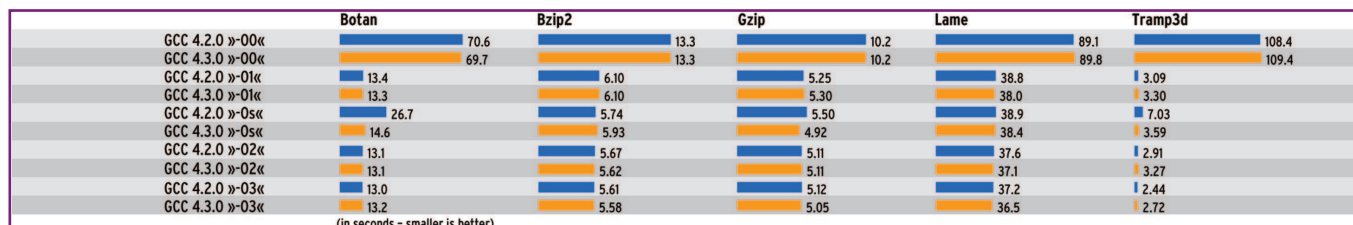


Figura 2: Cinco bancos de pruebas compilados con las dos versiones de GCC.

```
std::vector >
<std::vector<int> >
```

La sintaxis impide que el compilador identifique incorrectamente un operador de desplazamiento. En el futuro también funcionarán cosas como ésta:

```
std::vector >
<std::vector<int>>
```

En sistemas en producción son una buena noticia los tiempos de compilación más rápidos, ya que las inclusiones críticas han sido eliminadas de las cabeceras de la STL de C++. Esto podría significar que los desarrolladores tengan que incluir explícitamente las cabeceras como *limit.h*, *string.h* o *stdlib.h* en su código [7].

Los usuarios con CPUs de múltiples núcleos apreciarán que algunas clases de la STL y algoritmos puedan paralelizarse definiendo la macro *_GLIBCXX_PARALLEL*. Por otro lado, si utiliza las extensiones STL de GNU anteriores, como *hash_set* o *hash_map*, tendrá que enfrentarse al hecho de que G++ las eliminará próximamente; el estándar C++0x prevé por el contrario *tr1/unordered_set*, *tr1/hash_set* y similares.

Funciones Inline

En el caso de las funciones inline, el nuevo GCC tiene en cuenta el crecimiento de la pila. Cuando las ejecuciones de prueba se realizan con objeto de la optimización, el compilador de C utiliza ahora los tamaños de bloque de las operaciones de cadena como *memcpy()*, *memset()* y *bzero()* para crear código para bloques particularmente pequeños. Las operaciones *memcpy()* y

memset() han sido rediseñadas para que GCC haga actualmente uso del mejor algoritmo, dependiendo del tamaño del bloque y de la CPU de destino.

C++ y las emulaciones orientadas a objetos de C se benefician de una ejecución inline optimizada, particularmente para fragmentos inline tales como acceso por métodos *set()* y *get()* a propiedades en las que el código de la función es menor que el código que involucra su llamada. La vectorización automática se encuentra ahora activada por defecto para *-O3*, y se dice que es capaz de gestionar bucles complejos. Algunas optimizaciones nuevas reemplazan a los algoritmos de bajo rendimiento antiguos, reduciendo de nuevo los tiempos de compilación.

Una Clase Diferente de Café

Uno de los mayores cambios tienen que ver con el compilador GCJ Java, que los desarrolladores de GCC han reemplazado completamente por el compilador de Java de Eclipse.

Con este cambio radical Java 1.5 está completamente soportado, haciendo posible la creación de una pila completa de Java desde software libre con GCC y la rama Iced Tea Open JDK.

Algunas herramientas de Java, como *fastjar*, se han perdido debido al cambio, pero se proporciona *gjar* como sustituto. Otras, como *gcjh*, han sido rediseñadas completamente y no soportan el conjunto completo de argumentos soportado por las versiones previas.

Más Rápido

Para las pruebas de velocidad de *Linux Magazine* se ha utilizado un Apple Mac

Pro con un Intel Xeon, con velocidad de reloj de 3GHz y 8GB de RAM con un sistema Linux ejecutándose en modo x86 64-bit.

Las buenas noticias son que los tiempos de compilación están por debajo de los obtenidos por la versión previa. La Figura 1 muestra una comparativa entre GCC 4.2 y 4.3. El nuevo compilador es un poco más lento si se desactiva la optimización totalmente (con el parámetro *-O0*).

Los programas resultantes normalmente se ejecutan un poco más rápidos – el banco de pruebas de la Figura 2 muestra las mejoras tras el punto decimal. El paso atrás de la versión previa con respecto a la optimización del tamaño del programa (*-Os*) parece ser cosa del pasado, probablemente debido a los heurísticos de las funciones inline a las que se ha hecho referencia anteriormente.

RECURSOS

- [1] GCC: <http://gcc.gnu.org>
- [2] Cambios comparados con la versión previa: <http://gcc.gnu.org/gcc-4.3/changes.html>
- [3] Librería MPFR: <http://www.mpfr.org>
- [4] Direction flag manejada por el Kernel de Linux: <http://nvd.nist.gov/nvd.cfm?cvename=CVE-2008-1367>
- [5] Aritmética en Punto Flotante: <http://www2.hursley.ibm.com/decimal/>
- [6] Estado de C++ 200x (0x): http://gcc.gnu.org/gcc-4.3/cxx0x_status.html
- [7] Resumen de las modificaciones del código fuente de GCC 4.3: http://gcc.gnu.org/gcc-4.3/porting_to.html