Автор:
08.03.18 18:56 -

После шести месяцев разработки [представлен](#) релиз проекта [LLVM 6.0](#) (Low Level Virtual Machine) - GCC-совместимого инструментария (компиляторы, оптимизаторы и генераторы кода), компилирующего программы в промежуточный биткод RISC-подобных виртуальных инструкций (низкоуровневая виртуальная машина с многоуровневой системой оптимизации). Сгенерированный псевдокод может быть преобразован при помощи JIT-компилятора в машинные инструкции непосредственно в момент выполнения программы.

Напомним, что в соответствии с [новой нумерацией версий](#) осуществлён уход от разделения значительных и функциональных выпусков. В каждом функциональном обновлении теперь меняется первая цифра (например, осенью состоится релиз LLVM 7.0.0). Для обеспечения совместимости с существующими системами разбора номеров версий LLVM корректирующие обновления, как и раньше приводят к увеличению третьей цифры (6.0.1, 6.0.2, 6.0.3).

Из новых возможностей LLVM 6.0 отмечается включение в Clang по умолчанию стандарта C++14 ("-std=gnu++14" вместо "-std=gnu++98"), обеспечение поддержки некоторых возможностей будущего стандарта C++2a, интеграция патчей retpoline для блокирования второго варианта уязвимости Spectre, значительное улучшение поддержки отладочной информации CodeView для Windows, включение по умолчанию фреймворка GlobalISel для архитектуры AArch64 при сборке с уровнем оптимизации "-O0", добавление новых предупреждений компилятора.

[Улучшения](#) в Clang 6.0:

- В качестве диалекта языка C++ по умолчанию установлен gnu++14 ("-std=gnu++14") вместо ранее применявшегося gnu++98 ("-std=gnu++98"), что позволяет по умолчанию компилировать код, в котором присутствуют возможности, определённые в стандарте [C++14](#), включая специфичные расширения GNU;
- Добавлены некоторые возможности, развиваемые для будущего стандарта C++20 (кодовое название C++2a):
- Макрос __VA_OPT__ для адаптивного раскрытия вариативных макросов в зависимости от наличия токенов в вариативном аргументе;
- Поддержка оператора "‹=›" для трехстороннего сравнения;
- Поддержка инициализаторов элементов по умолчанию для битовых полей;
- Возможность лямбда-захвата выражений "*this";

- Вызов элементов по указателю (Pointer-to-member), используя определённые через выражение "const &" указатели на временные объекты;
- Оператор delete с деструктором, описанный в документе P0722R1 , но пока не одобренный для включения в спецификацию C++2a;

Для включения поддержки C++2a следует использовать опцию "-std=c++2a", при этом все указанные возможности, кроме __VA_OPT__ и "‹=›", доступны и в других режимах C++, но приводят к выводу предупреждения;

- Для блокирования в приложениях второго варианта уязвимости Spectre (CVE-2017-5715) в состав включён механизм Retpoline , основанный на применении специальной последовательности инструкций, исключающей вовлечение механизма спекулятивного выполнения для косвенных переходов (использование инструкции RET вместо JMP). Включение защиты осуществляется при помощи флага "-mretpoline" или "-mretpoline-external-thunk" при необходимости более тонкой настройки работы Retpoline;

- Добавлена поддержка конфигурационных файлов, в которых можно размещать коллекции из применяемых при сборке опций. Для использования настроек из файла конфигурации можно использовать опцию "--config foo.cfg" или создать исполняемый файл вида "foo-clang". Перечисленные в файле конфигурации опции включаются до опций, перечисленных в командной строке. Основным назначением файлов конфигурации является упрощение организации кросс-компиляции;
- Добавлены новые флаги "-fdouble-square-bracket-attributes" и "-fno-double-square-bracket-attributes" для включения и выключения атрибутов , задаваемых в двойных квадратных скобках, в любых языковых режимах;
- Для обеспечения совместимости с GCC добавлены языковые режимы "-std=c17", "-std=gnu17" и "-std=iso9899:2017" для включения поддержки нового стандарта языка Си. Режимы c17 и c11 отличаются только значением макроса __STDC_VERSION__, так как в новой спецификации отмечается только исправление ошибок;
- Добавлены флаги "-fexperimental-isel" и "-fno-experimental-isel" для включения и выключения нового фреймворка выбора инструкций GlobalISel . Данный фреймворк включен по умолчанию для архитектуры AArch64 при установке уровня оптимизации "-O0";
- Добавлен флаг "-nostdlib++" для отключения связывания со стандартной библиотекой C++ (действие аналогично вызову clang вместо clang++, но не отключает "-lm");
- Большая часть атрибутов Clang теперь доступна как в нотации GNU (__attribute((name))), так и в нотации Clang (). Добавлен встроенный макрос препроцессора __has_c_attribute() для динамической проверки доступности в режиме Си атрибутов, задаваемых в двойных квадратных скобках (данный синтаксис атрибутов включается флагом "-fdouble-square-bracket-attributes");
- Добавлена начальная поддержка сборки для платформы Windows на системах ARM64;

- Расширены возможности, связанные с поддержкой OpenCL и OpenMP;
- Прекращена поддержка операционной системы Bitrig , так как данный форк воссоединился с OpenBSD;
- Для обеспечения совместимости с заголовочными файлами стандартной библиотеки Visual Studio 2015 и 2017 значение _MSC_VER по умолчанию изменено с 1800 до 1911;
- По умолчанию в исполняемые файлы теперь добавляется секция ".init_array", если в системе не установлен GCC. Если наличие GCC обнаружено и версия старее 4.7.0, то как и раньше подставляется секция ".ctors";
- Добавлены новые встроенные макросы препроцессора __is_target_arch, __is_target_vendor, __is_target_os и __is_target_environment, которые могут применяться для определения отдельных характеристик целевой платформы;
- Расширены возможности для диагностики. Например, добавлена опция "-Wpragma-pack" для выявления проблем, возникающих при использовании директивы "#pragma pack", и опция "-Wtautological-constant-compare" для информирования о сравнении целочисленной переменной и целочисленной константой того же типа, имеющей наименьшее или наиболее из возможных для данного типа значений;
- В clang-format добавлена опция IndentPPDirectives для расстановки отступов для директив препроцессора, содержащих условные выражения (if/endif), а также опция IncludeBlocks для перегруппировки блоков заголовочных файлов;
- В статическом анализаторе обеспечено корректное определение и диагностика применения унарных операторов инкремента/декремента над неинициализированным значением;
- В linter clang-tidy добавлена большая порция новых проверок и обеспечена поддержка стандарта кодирования High Integrity C++ Coding Standard ;
- Добавлен минимальный runtime для компонента UBSan (Undefined Behavior Sanitizer) с реализацией детектора неопределенного поведения , выявляющего во время выполнения программы ситуации, когда поведение программы становится неопределенным. Runtime предоставляет только простые функции ведения лога событий во время выполнения и дедупликацию выводимых в лог записей ("-fsanitize=vptr" не поддерживается).

Основные новшества LLVM 6.0:

- Для архитектуры x86 добавлена поддержка CPU Intel Icelake и предоставляемых процессорами Intel расширенных инструкций VAES, GFNI, VPCLMULQDQ, AVX512VBMI2, AVX512BITALG и AVX512VNNI. Добавлена информация для планирования инструкций для процессоров Intel Sandy Bridge, Ivy Bridge, Haswell, Broadwell и Skylake. Улучшена модель планировщика для CPU AMD Jaguar. Улучшена генерация кода при сравнении областей памяти, умножении векторов i32, ротации целых векторов (XOP и AVX512), вычисления абсолютных скалярных целых значений, усечении векторов;
- В компоновщике LLD продолжено решение проблем с совместимостью, улучшена

поддержка форматов ELF и COFF, обеспечена возможность генерации компактной динамической relocation-таблицы в стиле Android, добавлена начальная поддержка WebAssembly (добавлен компоновщик wasm-ld), добавлены новые опции: --icf=none -z muldefs --plugin-opt --no-eh-frame-hdr --no-gdb-index --orphan-handling={place,discard,warn,error} --pack-dyn-relocs={none,android} --no-omagic --no-print-gc-sections --ignore-function-address-equality -z retpolineplt --print-icf-sections --no-pie;

- Добавлена предварительная поддержка санитайзеров (ASan, UBsan, TSan, MSan, SafeStack, libFuzzer) для платформы NetBSD на системах с архитектурой x86(_64);
- Значительно улучшено качество предоставления отладочной информации CodeView для Windows;
- Для архитектур x86 и ARM добавлена поддержка включения обработки исключений SjLj (setjmp/longjmp) на платформах, в которых SjLj не применяется по умолчанию;
- Внесены многочисленные улучшения в бэкенды для архитектур AArch64, ARM, AVR, Hexagon, MIPS и PowerPC. В том числе добавлена полная поддержка MIPS MT ASE, проведена работа по сокращению кода microMIPS, для AArch64 в режиме "-O0" включён по умолчанию фреймворк выбора инструкций GlobalISel.

Read more http://www.opennet.ru/opennews/art.shtml?num=48223