Вперед Назад Содержание

2. Командные Опции GNU CC

Когда вы вызываете GNU CC, он обычно выполняет препроцессирование, компиляцию, ассемблирование и линковку. "Общие опции" позволяют вам остановить этот процесс на промежуточной стадии. Например, опция '-c' говорит не запускать линкер. Тогда вывод состоит их объектных файлов, порожденных ассемблером.

Другие опции передаются на одну из стадий обработки. Одни опции управляют препроцессором, другие самим компилятором. Все еще имеются опции, управляющие ассемблером и линкером; большинство из них не документировано здесь, поскольку вам редко требуется использовать какую-нибудь из них.

Большая часть опций командной строки, которые вы можете использовать с GNU CC полезны для C программ; если опция полезна только для других языков (обычно C++), в объяснениии сказано об этом прямо. Если в описании какой-либо опции не упоминается исходный язык, вы можете использовать эту опцию со всеми поддерживаемыми языками.

См. раздел [Компиляция C++ программ], чтобы найти сводку опций для компиляции C++ программ.

Программа gcc принимает опции и имена файлов как операнды. Многие опции имеют многобуквенные имена; следовательно, многочисленные однобуквенные опции не могут быть сгруппированны: '-dr' очень отличается от '-d -r'.

Вы можете смешивать опции и другие аргументы. По большей части, используемый порядок не имеет значения. Порядок важен, когда вы используете несколько опций одного вида; например, если вы указываете '-L' больше чем один раз, директории просматриваются в порядке указания.

Многие опции имеют длинные имена, начинающиеся с '-f' или с '-W' - например, '-fforce-mem', '-fstrength-reduce', '-Wformat' и так далее. Большинство из них имеет положительную и отрицательную формы; отрицательной формой '-ffoo' будет '-fno-foo'. Это

 # Главная
 # О библиотеке

 # Выбор дистрибутива
преимущества Linux/UNIX | основные дистрибутивы | серверный Linux | BSD | LiveCDs | прочее

 # Установка и удаление программ
общие вопросы | каталоги софта | специальные случаи

 # Настройка и работа
установка, загрузчики | настройка Linux | консоль | файловые системы | процессы | шеллы, русификация, коммандеры | виртуальные машины, эмуляторы

 # X Window и оконные менеджеры
настройка X Window | GNOME | KDE | IceWM и др.

 # Работа с текстами
редакторы | офис | шрифты, кодировки и русификация | преобразования текстовых файлов | LaTeX, SGML и др. | словари

 # Графика
GIMP | фото | обработка изображений | форматы графических файлов

 # Сети, администрирование
общие вопросы | Dialup & PPP | брандмауэры | маршрутизация | работа в Windows-сетях | веб-серверы | Apache | прокси-серверы | сетевая печать | прочее

 # Программирование
GCC & GNU make | программирование в UNIX | графические библиотеки | Tcl | Perl | PHP | Java & C# | СУБД | CVS | прочее

Ядро
Мультимедиа
Интернет
Почта
Безопасность
Железо
Разное

 # Linux HowTo (как сделать)
 # Книги и руководства
 # Материалы на английском языке

руководство документирует только одну из этих форм – ту, которая не принимается по умолчанию.

2.1 Сводка Опций

Здесь изложена сводка всех опций, сгруппированная по типу. Пояснения расположены в следующих разделах.

Общие Опции

См. раздел [Опции, Управляющие Видом Вывода].

```
-c  -S  -E  -o FILE  -pipe  -v  -язык
```

Опции языка C

См. раздел [Опции, Управляющие Диалектом C].

```
-ansi  -fallow-single-precision -fcond-mismatch  -fno-asm
-fno-builtin  -fsigned-bitfields  -fsigned-char
-funsigned-bitfields  -funsigned-char  -fwritable-strings
-traditional  -traditional-cpp  -trigraphs
```

Опции Предупреждений

См. раздел [Опции для Включения или Подавления Предупреждений].

```
-fsyntax-only  -pedantic  -pedantic-errors
-w  -W  -Wall  -Waggregate-return  -Wbad-function-cast
-Wcast-align -Wcast-qual  -Wchar-subscript  -Wcomment
-Wconversion -Wenum-clash  -Werror  -Wformat
-Wid-clash-LEN  -Wimplicit  -Wimport  -Winline
-Wlarger-than-LEN  -Wmissing-declarations
-Wmissing-prototypes  -Wnested-externs
-Wno-import  -Woverloaded-virtual -Wparentheses
-Wpointer-arith  -Wredundant-decls -Wreorder -Wreturn-type -Wshadow
-Wstrict-prototypes  -Wswitch  -Wsynth  -Wtemplate-debugging
-Wtraditional  -Wtrigraphs -Wuninitialized  -Wunused
-Wwrite-strings
```

Опции Отладки

См. раздел [Опции для Отладки Вашей Программы или GNU CC].

```
-a  -dбуквы  -fpretend-float
-g  -gуровень -gcoff  -gdwarf  -gdwarf+
-ggdb  -gstabs  -gstabs+  -gxcoff  -gxcoff+
-p  -pg  -print-file-name=библиотека  -print-libgcc-file-name
-print-prog-name=программа  -print-search-dirs  -save-temps
```

Опции Оптимизации

См. раздел [Опции, которые Управляют Оптимизацией].

```
-fcaller-saves  -fcse-follow-jumps  -fcse-skip-blocks
-fdelayed-branch  -fexpensive-optimizations
-ffast-math  -ffloat-store  -fforce-addr  -fforce-mem
-finline-functions  -fkeep-inline-functions
-fno-default-inline  -fno-defer-pop  -fno-function-cse
-fno-inline  -fno-peephole  -fomit-frame-pointer
-frerun-cse-after-loop  -fschedule-insns
-fschedule-insns2  -fstrength-reduce  -fthread-jumps
-funroll-all-loops  -funroll-loops
```

-O -O0 -O1 -O2 -O3

Опции Препроцессора

См. раздел [Опции, Управляющие Препроцессором].

```
-Авопрос(ответ)  -C  -dD  -dM  -dN
-Dмакрос[=значение]  -E  -H
-idirafter директорий
-include файл  -imacros файл
-iprefix файл  -iwithprefix директорий
-iwithprefixbefore директорий  -isystem директорий
-M  -MD  -MM  -MMD  -MG  -nostdinc  -P  -trigraphs
-undef  -Uмакрос  -Wp,опция
```

Опции Ассемблера

См. раздел [Передача Опции Ассемблеру].

```
-Wa,опция
```

Опции Линкера

См. раздел [Опции Линковки].

```
имя-объектного-файла  -lбиблиотека
-nostartfiles  -nodefaultlibs  -nostdlib
-s  -static  -shared  -symbolic
-Wl,опция  -Xlinker опция
-u символ
```

Опции Директориев

См. раздел [Опции для Поиска в Директориях].

```
-Впрефикс  -Iдиректорий  -I-  -Lдиректорий
```

Целевые Опции

См. раздел [Указание Целевой Машины и Версии Компилятора].

```
-b машина  -V версия
```

Машинозависимые Опции

См. раздел [Модели и Конфигурации Машин].

Опции i386

```
-m486  -m386 -mieee-fp  -mno-fancy-math-387
-mno-fp-ret-in-387  -msoft-float  -msvr3-shlib
-mno-wide-multiply -mrtd -malign-double
-mreg-alloc=список -mregparm=число
-malign-jumps=число -malign-loops=число
-malign-functions=число
```

Опции Генерации Кода

См. раздел [Опции Соглашений о Генерации Кода].

```
-fcall-saved-регистр  -fcall-used-регистр
-ffixed-регистр  -finhibit-size-directive
-fno-common  -fno-ident  -fno-gnu-linker
-fpcc-struct-return  -fpic  -fPIC
-freg-struct-return  -fshared-data  -fshort-enums
-fshort-double  -fvolatile  -fvolatile-global
-fverbose-asm -fpack-struct +e0  +e1
```

2.2 Опции, Управляющие Видом Вывода

Компиляция может включать до четырех стадий: препроцессирование, собственно компиляцию, ассемблирование и линковку, всегда в этом порядке. Первые три стадии применяются к отдельному исходному файлу и заканчиваются получением объектного файла; линковка объединяет все объектные файлы (заново откомпилированные или полученные как входные) в исполняемый файл.

Для любого имени входного файла суффикс определяет какая компиляция требуется:

file.c	Исходный код на C, который нуждается в препроцессировании.
file.i	Исходный код на C, который не нуждается в препроцессировании.
file.ii	Исходный код на C++, который не нуждается в препроцессировании.
file.m	Исходный код на Objective C. Заметим, что вам необходимо подключить библиотеку 'libobjc.a', чтобы заставить Objective C программу работать.
file.h	C заголовочный файл (не для компиляции или линковки).
file.cc	
file.cxx	
file.cpp	
file.C	Исходный код на C+, который нуждается в препроцессировании.
file.s	Ассемблерный код.
file.S	Ассемблерный код, который нуждается в препроцессировании.
другие	Объектный файл, который нужно отдать прямо на линковку. Так поступают с любым именем файла с нераспознанным суффиксом.

Вы можете прямо указать входной язык при помощи опции '-x':

-x язык.

Прямо специфицирует язык последующих входных файлов (даже если компилятор может выбрать язык на основании суффикса имени файла). Эта опция действует на все

входные файлы вплоть до следующего появления опции '-x'. Возможными значениями для языка являются:

```
c objective-c c++
c-header  cpp-output  c++-cpp-output
assembler assembler-with-cpp
```

-x none

Выключает любое указание языка так, что последующие файлы обрабатываются в соответствии с суффиксами имен файлов (как если бы '-x' не указывалось бы вовсе).

Если вам нужны лишь некоторые из стадий компиляции, вы можете использовать '-x', чтобы указать gcc где начать, и одну из опций '-c', '-S' или '-E', чтобы указать, где gcc должен остановиться. Заметим, что некоторые комбинации (например, '-x cpp-output -E' указывают gcc ни делать вообще ничего).

-c

Компилировать или ассемблировать исходные файлы, но не линковать. Стадия ликовки просто не выполняется. Конечный вывод происходит в форме объектного файла для каждого исходного файла.

По умолчанию, имя объектного файла делается из имени исходного файла заменой суффикса '.c', '.i', '.s', и.т.д. на '.o'.

Нераспознанные входные файлы, не требующие компиляции или ассемблирования, игнорируются.

-S

Остановиться после собственно компиляции; не ассемблировать. Вывод производится в форме файла с ассемблерным кодом для каждого не ассемблерного входного файла.

По умолчанию, имя файла с ассемблерным кодом делается из имени исходного файла заменой суффикса '.c', '.i', и.т.д. на '.s'.

Входные файлы, которые не требуют компиляции игнорируются.

-E

Остановиться после стадии препроцессирования; не запускать собственно компилятор. Вывод делается в форме препроцессированного исходного кода, который посылается на стандартный вывод.

Входные файлы, которые не требуют препроцессирования игнорируются.

-o файл

Поместить вывод в файл 'файл'. Эта опция применяется вне зависимости от вида порождаемого файла, есть ли это выполнимый файл, объектный файл, ассемблерный файл или препроцессированный C код.

Поскольку указывается только один выходной файл, нет смысла использовать '-o' при компиляции более чем одного входного файла, если вы не порождаете на выходе выполнимый файл.

Если '-o' не указано, по умолчанию выполнимый файл помещается в 'a.out', объектный файл для 'исходный.суффикс' - в 'исходный.o', его ассемблерный код в 'исходный.s' и все препроцессированные C файлы - в стандартный вывод.

-v

Печатать (в стандартный вывод ошибок) команды выполняемые для запуска стадий компиляции. Также печатать номер версии управляющей программы компилятора, препроцессора и самого компилятора.

-pipe

Использовать каналы вместо временных файлов для коммуникации между различными стадиями компиляции. Это может не работать на некоторых системах, где ассемблер не может читать из канала, но ассемблер GNU не имеет проблем.

2.3 Опции, Управляющие Диалектом C

Следующие опции управляют диалектами C (или или языков порожденных из C, таких как C++ и Objective C), которые воспринимает компилятор:

-ansi

Поддерживает все ANSI C программы.

Эта опция выключает некоторые свойства GNU CC, которые несовместимы с ANSI C такие, как ключевые слова asm, inline и typeof, предопределенные макросы такие, как unix и vax, которые идентифицируют тип используемой вами системы. Она также включает нежелательные и редко используемые ANSI трехсимвольные последовательности, не разрешает '$' в идентификаторах и комментарии в стиле C++ '//'.

Альтернативные ключевые слова __asm__, __extension__, __inline__ и __typeof__ продолжают работать не смотря на '-ansi'. Вы, разумеется, не хотели бы использовать их в ANSI C программе, но полезно использовать их в заголовочных файлах, которые могут быть включены в компиляции с '-ansi'.

Опция '-ansi' не действует так, что не ANSI C программы беспричинно отбрасываются. Чтобы это было так, в добавление к '-ansi' требуется опция '-pedantic'. См. раздел [Опции Предупреждений], стр. 35.

Когда используется опция '-ansi', предопределен макрос __STRICT_ANSI__. Некоторые заголовочные файлы могут заметить этот макрос и воздерживаться от объявления определенных функций или определения определенных макросов, к которым ANSI стандарт не обращается; это делается, чтобы избежать помех программам, которые могут использовать эти имена для других целей.

Функции alloca, abort, exit и _exit не являются встроенными функциями при использовании '-ansi'.

-fno-asm

Не распознаются как ключевые слова asm, inline и typeof, и они могут быть использованы в коде в качестве идентификаторов. Вы можете использовать вместо них ключевые слова __asm__, __inline__ и __typeof__. '-ansi' включает '-fno-asm'.

-fno-builtin

Не распознаются встроенные функции кроме тех, имена которых начинаются с двух подчеркиваний. На данный момент затрагиваемыми функциями являются abort, abs, alloca, cos, exit, fabs, ffs, labs, memcmp, memcpy, sin, sqrt, strcmp, strcpy и strlen.

GCC обычно генерирует специальный код, чтобы более эффективно обрабатывать некоторые встроенные функции; например, вызов alloca может стать просто последовательностью инструкций, которые прямо выравнивают стек, а вызов memcpy может превратиться просто в цикл копирования. Результирующий код является и более коротким, и более быстрым, но поскольку вызова функции как такового больше не существует, вы не можете ни поставить на него точку останова, ни изменить поведение функции прилинковав другую библиотеку.

Опция '-ansi' не дает функциям alloca и ffs быть встроенными, поскольку эти функции не имеют предопределенного значения в стандарте ANSI C.

-trigraphs

Поддерживаются трехсимвольные последовательности ANSI C. Опция '-ansi' включает '-trigraphs'. Не забивайте себе голову этим сумасшедствием.

-traditional

Пытается поддержать некоторые свойства традиционных компиляторов C. А именно:

- Все extern объявления имеют глобальное действие, даже если они написаны внутри определения функции. Это распространяется и на точные объявления функций.
- Более новые ключевые слова typeof, inline, signed, const и volatile не распознаются. (Вы все еще можете использовать альтернативные ключевые слова такие, как __typeof__, __inline__ и т.д.)
- Всегда разрешены сравнения между целыми и указателями.
- Целые типы unsigned short и unsigned char расширяются к unsigned int.
- Константы с плавающей точкой вне диапозона не являются ошибками.

- Некоторые конструкции, которые ANSI считает одним неправильным числом, такие как '0xe-0xd', считаются выражениями.
- Строковые "константы" не обязательно константны; они размещаются в памяти доступной для записи, и одинаково выглядящие константы размещаются в разных местах. (Это тот же действие, что и у '-fwritable-strings'.)
- Все автоматические переменные не объявленные register сохраняются при longjump'e. Обычно GNU C следует ANSI C: автоматические переменные не объявленные volatile могут затираться.
- Символьные последовательности '\x' и '\a' воспринимаются как 'x' и 'a' соответственно. Без '-traditional', '\x' является префиксом для шестнадцатеричного представления символа, а '\a' порождает символ звонка.

Вы можете захотеть использовать 'fno-builtin' так же как и '-traditional', если ваша программа использует имена, которые обычно являются именами встроенных функций GNU C для его собственных целей.

Вы не можете использовать '-traditional', если вы включаете какие либо заголовочные файлы, которые полагаются на особенности ANSI C. Некоторые продавцы поставляют системы с ANSI C заголовочными файлами и вы не можете использовать '-traditional' на таких машинах для компиляции файлов, который включают системные заголовочные файлы.

В препроцессоре коментарии преобразуются в ничто, а не в пробел. Это разрешает традиционное объединение лексем.

В директиве препроцессора символ '#' должен быть первым в строке.

В препроцессоре аргументы макроса распознаются в строковых константах в определении макроса (их значения превращаются в строки, но без дополнительных кавычек, когда они появляются в соответствующем контексте).

Предопределенный макрос __STDC__ не определен, если вы используете '-traditional', но __GNUC__ определен (поскольку расширения GNU, о которых говорит __GNUC__ не затрагиваются опцией '-traditional'). Если вам нужно написать заголовочный файл, который работает по-разному в зависимости от того, используется ли опция '-traditional', проверив эти два макроса вы можете различить четыре ситуации: GNU C, традиционный GNU C, другие ANSI C компиляторы и другие компиляторы старого C.

Предопределенный макрос __STDC_VERSION__ также не определен, когда вы используете '-traditional'. См. раздел "Стандартные Предопределенные Макросы" в 'Препроцессоре C' по поводу этих и других предопределенных макросов.

Препроцессор считает, что строковые константы всегда заканчиваются при переходе на новую строку (если только ему не предшествует '\'). (Без '-traditional' в строковых константах может быть символ новой строки так, как набрано.)

-traditional-cpp

Пытается поддержать некоторые особенности традиционных препроцессоров C. Это включает в себя пять пунктов в таблице непосредственно выше, но без других эффектов '-traditional'.)

-fcond-mismatch

Позволяются условные выражения с не согласованными типами второго и третьего аргументов. Значение такого выражения есть void.

-funsigned-char

Тип char считается беззнаковым, как unsigned char.

Каждая машина имеет умолчание – каким должен быть char, таким, как unsigned char или таким, как signed char.

В идеале, переносимая программа всегда должна использовать unsigned char или signed char, когда она зависит от знаковости объекта. Но многие программы были написаны с использованием простого char, в предположении, что он будет либо знаковым, либо беззнаковым, в зависимости от машины, для которой они были написаны. Эта опция и ей противоположная позволяют вам заставить такую программу работать при противоположном умолчании.

Тип char всегда отличен и от signed char, и от unsigned char, хотя его поведение всегда такое же, как у одного из этих двух типов.

-fsigned-char

Тип char считается знаковым, как signed char.

Заметим, что эта опция эквивалентна '-fno-unsigned-char', которая является отрицательной формой '-funsigned-char'. Анологично, опция '-fno-signed-char' эквивалентна '-funsigned-char'.

-fsigned-bitfields

-funsigned-bitfields

-fno-signed-bitfields

-fno-unsigned-bitfields

Эти опции управляют знаковостью поля битов, в том случае, если его объявление не использует signed или unsigned. По умолчанию такие поля считаются знаковыми, потому что это последовательно: базовые целые типы такие, как int являются знаковыми.

Однако, если используется опция '-traditional', битовые поля все беззнаковые - не важно какие.

-fwritable-strings

Строковые константы сохраняются в сегменте, доступном для записи и не сливаются. Это нужно для совместимости со старыми программами, которые предполагают, что можно писать в строковую константу. Опция '-traditional' также имеет такой эффект.

Запись в строковые константы это очень плохая идея - "константы" должны быть константами.

-fallow-single-precision

Числа с плавающей точкой одинарной точности не расширяются до двойной при математических операциях, даже при компиляции с опцией '-traditional'. Классический C Кернигана и Риттчи расширяет все операции с плавающей точкой до двойной точности, вне зависимости от размеров операндов. На архитектуре, для которой вы компилируете одинарная точность может оказаться быстрее двойной. Если

вы должны использовать '-traditional', но хотите использовать операции одинарной точности, когда операнды одинарной точности, используйте эту опцию. Эта опция не имеет эффекта при компиляции с ANSI или GNU C соглашениями.

2.4 Опции для Включения или Подавления Предупреждений

Предупреждения - это диагностические сообщения, которые сообщают о конструкциях не являющихся заведомо ошибочными, но рискованных или, вероятно, содержащими ошибку.

Вы можете включить выдачу многих специфических предупреждений с помощью опций, начинающихся с '-W', например, '-Wimplicit', чтобы включить предупреждения на неявных объявлениях. Каждая из таких опций имеет отрицательную форму, которая начинается с '-Wno-', и служит для подавления выдачи предупреждений; например, 'Wno-implicit'. В этом руководстве указана только одна форма каждой опции - та, которая не является умолчанием.

Следующие опции управляют количеством и видом предупреждений, выдаваемых GNU CC.

-fsyntax-only

Проверяется код на наличие синтаксических ошибок, но после этого не делается ничего.

-pedantic

Выдаются все предупреждения, требуемые строгим ANSI стандартом C, отбрасываются все программы, которые используют запрещенные расширения.

Правильные ANSI C программы корректно компилируются как с, так и без этой опции (хотя очень редкие и требуют опции '-ansi'). Однако, без этой опции поддерживаются также некоторые GNU расширения и свойства традиционного C. С этой опцией они отклоняются.

'-pedantic' не вызывает предупреждения при использовании альтернативных ключевых слов, начинающихся и заканчивающихся с '__'. Предупреждения также не появляются после выражений, переж которыми стоит '__extension__'. Однако, этот исключительный

способ следует использовать только в системных заголовочных файлах; следует избегать его в прикладных программах. См. раздел 6.34 [Альтернативные Ключевые Слова].

Эта опция не претендует на полезность - она существует только чтобы удовлетворить педантов, которые в противном случае будут утверждать, что GNU CC не может поддерживать стандарт ANSI C.

Некоторые пользователи пытаются использовать '-pedantic', чтобы проверить программу на соответствие со строгим стандартом ANSI C. Они часто обнаруживают, что она делает не ровно то, что они хотят: она выявляет некоторые не ANSI C конструкции, но не все, а только те, для которых ANSI C требует диагностику.

Возможность выявить все несоответствия с ANSI C могла бы быть полезна во многих случаях, но потребовала бы гораздо больше дополнительной работы совершенно отличной от целей опции '-pedantic'. Мы скорее рекомендуем пользователю воспользоваться преимуществами расширений GNU C и игнорировать ограничения других компиляторов. Кроме некоторых суперкомпьютеров и крайне маленьких машин, есть весьма мало причин пользоваться каким либо другим компилятором C вместо раскручиваемого компилятора GNU CC.

-pedantic-errors

Подобна опции '-pedantic', с тем отличием, что вместо предупреждений порождаются ошибки.

-w

Отменяются все предупреждения.

-Wno-import

Отменяются предупреждения об использовании '#import'.

-Wchar-subscripts

Предупреждает, если индекс массива имеет тип char. Это частая причина ошибок, так как программисты часто забывают, что этот тип может быть знаковым на некоторых

машинах.

-Wcomment

Предупреждает при каждом появлении начала комментария '/*' в комментарии.

-Wformat

Проверяет вызовы к printf, scanf и т. д., чтобы проверить типы передаваемых параметров на соответствие со специфицированной строкой формата.

-Wimplicit

Предупреждает, когда функция или параметр объявляются неявно.

-Wparentheses

Предупреждает об отсутствии скобок в некоторых контекстах, таких как присваивание там, где нужно значение или вложенные операции, в приоритетах которых люди часто путаются.

-Wreturn-type

Предупреждает, если функция объявлена с типом возвращаемого значения по умолчанию - int. Также предупреждает об операторах возврата без возвращаемого значения в функциях, чей возвращаемый тип не есть void.

-Wswitch

Предупреждает, когда оператор switch имеет в качестве переключателя тип перечисления, и для одного или нескольких именованных кодов отсутствует case. (Присутствие метки default подавляет это предупреждение.) Метка case вне диапозона перечисления также вызывает прежупреждение при использовании этой опции.

-Wtrigraphs

Предупреждает, если встречается какая-либо трехсимвольная ANSI C последовательность (предполагается, что они включены).

-Wunused

Предупреждает, если переменная не используется вне ее описания, или если функция объявлена static, но нигде не определена, или если метка объявлена, но не используется, или если оператор вычисляет значение, которое точно нигде не используется.

Чтобы подавить выдачу такого предупреждения на выражении просто поставьте перед ним приведение к void. Для не используемых переменных и параметров используйте атрибут 'unused' (см. раздел 6.28 [Атрибуты Переменных]).

-Wuninitialized

Автоматическая переменная используется без инициализации.

Это предупреждение возможно только в оптимизирующей компиляции, поскольку требуемый анализ потока данных выполняется только при оптимизации. Если вы не используете опцию '-O', вы просто не получите эти предупреждения.

Эти предупреждения выдаются только для переменных, которые являются кандидатами на выделение регистров. Следовательно, они не выдаются для тех переменных, которые объявлены volatile, или тех, чей адрес берется, или чей размер отличен от 1, 2, 4 или 8 байт. Они, также, не выдаются для структур, объединений и массивов, даже если они размещаются на регистрах.

Заметим, что может не быть предупреждения о переменной, которая используется только чтобы вычислить значение, которое более не используется, потому что это вычисление может быть удалено анализом потока данных раньше, чем печатаются предупреждения.

Эти предупреждения сделаны необязательными, потому что GNU CC недостаточно умен, чтобы видеть все случаи, при котрых код может быть правильным, несмотри на кажущуюся ошибку. Ниже показан один пример, как такое может случиться:

```
{
  int x;
  switch (y)
    {
```

```
      case 1: x = 1;
        break;
      case 2: x = 4:
        break;
      case 3: x = 5;
      }
    foo (x);
}
```

Если значение y всегда есть 1, 2 или 3, тогда x всегда иниацализирована, но GNU CC не знает этого. Ниже показан другой общий случай:

```
{
  int save_y;
  if (change_y) save_y = y, y = new_y;
  ...
  if (change_y) y = save_y;
}
```

Здесь нет ошибки, потому что save_y используется, только если она была установлена.

Некоторые ложные предупреждения могут быть устранены, если вы объявите все используемые функции, которые никогда не возвращают управление, как noreturn. См. раздел 6.22 [Атрибуты Функций].

-Wall

Все вышеперечисленные опции '-W' вместе взятые.

Остающиеся опции '-W...' не включаются в '-Wall', потому что они предупреждают о конструкциях, которые мы считаем разумным использовать при случае в хороших программах.

-W

Печатает дополнительные предупреждения при следующих случаях:

- Не volatile автоматическая переменнная может измениться при longjump'e. Эта опция также возможна только при оптимизирующей компиляции. Компилятор смотрит

только на вызовы setjump. Он не может знать, где будет вызвана longjump; в действительности, обработчик сигнала может вызвать ее в любом месте кода. В результате вы можете получить предупреждение, даже если на самом деле нет никаких проблем, потому что longjump вызывается в месте, которое не порождает проблемы.

- Функция может как возвращать, так и не возвращать значение. (Попадание в конец тела функции рассматривается как возврат без значения.) К примеру, следующяя функция вызывает данное предупреждение:

```
foo (a)
{
  if (a > 0)
    return a;
}
```

- Выражение в левой части операции ',' не содержит побочных эффектов. Чтобы подавить выдачу этого предупреждения, приведите неиспользуемое выражение к void. Например, выражение x[i,j] вызовет данное предупреждение, а выражение x[(void)i,j] - нет.
- Беззнаковое значение сравнивается с нулем с помощью '<' или '<='.
- Появляется выражение сравнения вида 'x<=y<=z'; оно эквивалентно '(x<=y ? 1 : 0) <= z', которое отлично от интерпретации обычной математической нотации.
- Класс памяти вроде static не является первым словом в объявлении. В соответствии со стандартом C такое использование является устаревшим.
- Если указана также опция '-Wall' или '-Wunused', предупреждает о не используемых аргументах.
- Агрегат имеет инициализатор с частью скобок. Например, следующий код вызывает данное предупреждение из-за отсутствия скобок вокруг инициализатора x.h: struct s { int f, g; }; struct t { struct s h; int i; }; struct t x = { 1, 2, 3 };

-Wtraditional

Предупреждает об определенных конструкциях, которые ведут себя неодинаково в классическом и ANSI C.

- Макроаргумент в строковой константе, появившийся в теле макроса. Это вызывает подстановку аргумента в классическом C, но в ANSI C аргумент остается частью строки.
- Функция объявлена внешней в блоке и использована после конца блока.
- Оператор switch имеет операнд типа long.

-Wshadow

Предупреждает, когда одна локальная переменная затеняет другую локальную переменную.

-Wid-clash-длина

Предупреждает, когда у двух различных идентификаторов совпадают первые символы в количестве, определяемым параметром "длина". Это может вам помочь подготовить программу к компиляции устаревшим, ненормальным компилятором.

-Wlager-than-длина

Предупреждает, когда объявляется объект размелом больше чем "длина" байт.

-Wpointer-arith

Предупреждает обо всем, что зависит от размеров типа функции и типа void. GNU C назначает этим типам размер 1 для удобства вычислений с указателями на void и на функции.

-Wbad-function-cast

Предупреждает, когда вызов функции приводится к несоответствующему типу. Например, вызов int malloc () приводится к указателю.

-Wcast-qual

Предупреждает, когда указатель приводится так, чтобы убрать квалификатор типа. Например, const char * приводится к обычному char *.

-Wcast-align

Предупреждает, если указатель приводится так, что возрастают требования на выравнивание. Например, если char * приводится к int * на машине, где int может размещаться только по двух- или четырехбайтовой границе.

-Wwrite-strings

Дает строковым константам тип const char[длина] так, что копирование адреса в не константный указатель на char вызовет предупреждение. Эти предупреждения помогут вам найти код времени компиляции, который может пытаться писать в строковую константу, но только в том случае, если вы будете внимательно относиться к использованию const в объявлениях и прототипах. В противном случае это превратится только в неприятности; вот почему мы не заставляем '-Wall' вызывать эти предупреждения.

-Wconversion

Предупреждает, если прототип вызывает преобразование типа отличное от того, которое было бы с тем же аргументом при отсутствии прототипа. Это включает преобразования целочисленных типов в плавающие и т. п., и преобразования, меняющие размер или знаковость целочисленного аргумента, кроме тех, которые совпадают с целочисленным расширением.

Также предупреждает, если отрицательная целая константа неявно приводится к беззнаковому типу. Например, предупреждает об присваивании x = -1, если x беззнаковое. Но не предупреждает об прямо указаном преобразовании типа (unsigned)-1.

-Wagregate-return

Предупреждает, если определяется или вызывается функция, которая возвращает структуру. (В языках, где можно возвращать массивы, это также вызывает предупреждение.)

-Wstrict-prototypes

Предупреждает, если функция объявлена или определена без спецификации типов аргументов. (Определение функции в старом стиле не вызывает предупреждения, если

перед ним было объявление с указанными типами аргументов.)

-Wmissing-prototypes

Предупреждает, если глобальная функция определена без предварительного объявления прототипа. Это предупреждение появляется, даже если само определение дает прототип. Цель этого предупреждения в том, чтобы выявить глобальные функции, которые не объявлены в заголовочных файлах.

-Wredundant-decls

Предупреждает, если нечто объявлено более чем один раз в одной области действия даже в тех местах, где многократные объявления допустимы и ничего не меняют.

-Wnested-externs

Предупреждает, если extern объявление встречается дважды внутри функции.

-Winline

Предупреждает, если функция не может быть сделана inline и при этом была объявлена inline, или же была дана опция '-finline-functions'.

-Werror

Превращает все предупреждения в ошибки.

2.5 Опции для Отладки Ваших Программ или GNU CC

GNU CC имеет различные специальные опции, которые используются для отладки как вашей программы, так и GCC.

-g

Порождает отладочную информацию в родном формате операционной системы (stabs, COFF, XCOFF или DWARF). GDB может работать с этой отладочной информацией.

В большинстве систем, которые используют формат stabs, '-g' включает использование дополнительной отладочной информации, которую может использовать только GDB; эта дополнительная отладочная информация делает работу отладки в GDB лучше, но может, вероятно испортить работу других отладчиков, или помешать им прочитать программу. Если вы хотите управлять тем, будет ли порождаться дополнительная информация, используйте '-gstabs+', '-gstabs', '-gxcoff+', '-gxcoff', '-gdwarf+', '-gdwarf' (см. ниже).

В отличие от большинства других компиляторов C, GNU CC позволяет использовать опцию '-g' вместе с '-O'. Сокращения кода сделанные при оптимизации могут случайно привести к неожиданным результатам: некоторые переменные, которые вы объявили могут не существовать вовсе; поток управления может переместиться, когда вы этого не ожидали; некоторые операторы могут не выполняться, потому что они производят постоянный результа, или же их результат уже известен; некоторые операторы могут выполняться в других местах, потому что их вынесли из циклов.

Тем не менее, GNU CC дает возможность отлаживать оптимизированный результат. Это делает разумным использование оптимизатора для отладки программ, которые могут содержать ошибки.

Следующие опции полезны, если GNU CC сгенерирован с возможностью использовать более чем один формат отладочной информации.

-ggdb

Порождает отладочную информацию в родном формате (если он поддерживается), включающую расширения GDB, если это вообще возможно.

-gstabs

Порождает отладочную информацию в формате stabs (если он поддерживается) без расширений GDB. Это формат, используемый DBX'ом в большинстве BSD систем. На MIPS, Alpha и в системе System V Release 4 эта опция порождает отладочную информацию в формате stabs, которую не понимает DBX или SDB. В системе System V Realease 4 эта опция требует ассемблера GNU.

-gstabs+

Порождает отладочную информацию в формате stabs (если он поддерживается) с использованием расширений GNU, которые понимает только отладчик GNU (GDB). Использование этих расширений с другими отладчиками вероятно приведет к их краху или помешает прочитать программу.

-gcoff

Порождает отладочную информацию в формате COFF (если он поддерживается). Этот формат использовался SDB в системах System V до System V Release 4.

-gxcoff

Порождает отладочную информацию в формате XCOFF (если он поддерживается). Это формат, используемый отладчиком DBX в системах IBM RS/6000.

-gxcoff+

Порождает отладочную информацию в формате XCOFF (если он поддерживается) с использованием расширений GNU, которые понимает только отладчик GNU (GDB). Использование этих расширений с другими отладчиками вероятно приведет к их краху или помешает прочитать программу, а также может привести ассемблер, отличный от ассемблера GNU (GAS), к ошибке.

-gdwarf

Порождает отладочную информацию в формате DWARF (если он поддерживается). Этот формат используется SDB в большинстве систем System V Release 4.

-gdwarf+

Порождает отладочную информацию в формате DWARF (если он поддерживается) с использованием расширений GNU, которые понимает только отладчик GNU (GDB). Использование этих расширений с другими отладчиками вероятно приведет к их краху или помешает прочитать программу.

-gуровень

-ggdbуровень

-gstabsуровень

-gcoffуровень

-gxcoffуровень

-gdwarfуровень

Включает отладочную информацию, а также использует 'уровень', чтобы определить как много информации требуется. По умолчанию используется уровень 2.

Уровень 1 выводит минимальную информацию, достаточную, чтобы отслеживать цепочку вызовов в частях программы, которые вы не собираетесь отлаживать. Она включает в себя описание функций и внешних переменных, но не содержит информации об локальных переменных и номерах строк.

Уровень 3 включает дополнительную информацию, такую как все макро определения встречающиеся в программе. Некоторые отладчики поддерживают макро расширения при использовании '-g3'.

-p

Порождает дополнительный код для записи профилирующей информации, подходящей для анализирующей программы prof. Вы должны использовать эту опцию при компиляции исходного файла, о котором вы хотите получить информацию, и вы также должны использовать ее при линковке.

-pg

Порождает дополнительный код для записи профилирующей информации, подходящей для анализирующей программы gprof. Вы должны использовать эту опцию при компиляции исходного файла, о котором вы хотите получить информацию, и вы также должны использовать ее при линковке.

-a

Порождает дополнительный код для записи профилирующей информации для базисных блоков, который будет записывать, сколько раз выполняется каждый базисный блок,

его начальный адрес и имя функции, которая его содержит. Если используется опция '-g', будут также записываться номер строки и имя файла начала базисного блока. Если не изменено описанием архитектуры машины, то действием по умолчанию является добавление к текстовому файлу 'bb.out'.

Эти данные могут быть проанализированы программой вроде tcov. Заметим, однако, что формат данных не таков, как ожидает tcov. GNU gprof со временем будет расширен для обработки этих данных.

-dбуквы

Делает отладочные дампы во время компиляции в моменты времени определяемые буквами. Они используются для отладки компилятора. Имена файлов для большинства дампов получаются из имени исходного файла добавлением слова (например, 'foo.c.rtl' или 'foo.c.jump'). Ниже указаны возможные буквы и их значения:

'M'

Дамп всех макро определений в конце препроцессирования; не порождать другого вывода.

'N'

Дамп всех макро определений в конце препроцессирования.

'D'

Дамп всех макро определений в конце препроцессирования, в добавление к нормалному выводу.

'y'

Дамп отладочной информации во время лексического разбора в стандартный вывод ошибок.

'r'

Дамп после генерации RTL в файл 'файл.rtl'.

'x'

Только генерировать RTL для функции вместо ее компиляции. Обычно используется вместе с 'r'.

'j'

Дамп после первой оптимизации переходов в файл 'файл.jump'.

's'

Дамп после CSE (включая оптимизацию переходов, которая иногда следует за CSE) в файл 'файл.cse'.

'L'

Дамп после оптимизации циклов в файл 'файл.loop'.

't'

Дамп после второго прохода CSE (включая оптимизацию переходов, которая иногда следует за CSE) в файл 'файл.cse2'.

'f'

Дамп после анализа потока в файл 'файл.flow'.

'c'

Дамп после комбинирования инструкций в файл 'файл.combine'.

'S'

Дамп после первого прохода планировщика в файл 'файл.sched'.

'l'

Дамп после локального распределения регистров в файл 'файл.lreg'.

'g'

Дамп после глобального распределения регистров в файл 'файл.greg'.

'R'

Дамп после второго прохода планировщика в файл 'файл.sched2'.

'J'

Дамп после последней оптимизации переходов в файл 'файл.jump2'.

'd'

Дамп после планировщика ветвей в файл 'файл.dbr'.

'k'

Дамп после преобразования регистров в стек в файл 'файл.stack'.

'a'

Порождает все дампы, перечисленные выше.

'm'

Печатает статистику использования памяти в конце исполнения в стандартный вывод ошибок.

'p'

Комментирует ассемблерный вывод комментариями, показывающими какой шаблон и какая альтернатива были использованы.

-fpretend-float

При запуске кросс-компилятора делает вид, что целевая машина использует тот же формат чисел с плавающей точкой, что и базовая машина. Это приводит к неверному выводу реыльных констант с плавающей точкой, но реальная последовательность

инструкций будет, вероятно, такой же как и последовательность, порождаемая GNU CC при запуске на целевой машине.

-save-temps

Постоянно сохраняет промежуточные "временные" файлы; они помещаются в текущий каталог, а их имена основываются на имени исходного файла. Так, компилируя 'foo.c' с опциями '-c -save-temps', будут порождены файлы 'foo.i', 'foo.s', также как и 'foo.o'.

-print-file-name=библиотека

Печатает полное абсолютное имя библиотечного файла 'библиотека', которое использовалось бы при линковке, и не делает больше ничего. С этой опцией GNU CC ничего не компилирует и не линкует - он только печатает имя файла.

-print-prog-name=программа

Подобна '-print-file-name', но ищет программу, такай как, например, 'cpp'.

-print-libgcc-file-name

Совпадает с '-print-file-name=libgcc.a'.

Эта опция полезна, когда вы используете '-nostdlib' или '-nodefaultlibs', но вы хотите линковать с 'libgcc.a'. Вы можете написать

```
gcc -nostdlib файлы... 'gcc -print-libgcc-file-name'.
```

-print-search-dirs

Печатает имя сконфигурированной инсталяционной директории, список программ и библиотечных директориев, которые будет просматривать gcc и не делает больше ничего.

Эта опция полезна, когда gcc печатает сообщение об ошибке 'installation problems, cannot exec cpp: No such file or directory' ('проблемы инсталяции, невозможно выполнить cpp: Нет такого файла или директория'). Чтобы разрешить эту проблему, вы

либо должны поместить 'cpp' и другие компоненты компилятора там, где gcc ожидает их найти, либо вы можете установить переменную окружения GCC_EXEC_PREFIX равной дериктории, где вы их поместили. Не забывайте сопутствующие символы '/'. См. раздел [Переменные Окружения].

2.6 Опции, которые Управляют Оптимизацией

Эти опции управляют различными видами оптимизаций:

-O

-O1

Оптимизировать. Оптимизированная трансляции требует несколько больше времени и несколько больше памяти для больших функций.

Без `-O' цель компилятора состоит в том, чтобы уменьшить стоимость трансляции и выдать ожидаемые результаты при отладке. Операторы независимы: если вы останавливаете программу на контрольной точке между операторами, вы можете назначить новое значение любой переменной или поставить счетчик команд на любой другой оператор в функции и получить точно такие результаты, которые вы ожидали из исходного текста.

Без `-O' компилятор только распределяет переменные объявленные register в регистрах. Получающийся в результате откомпилированный код немного хуже чем произведенный PCC без `-O'.

С `-O' компилятор пробует уменьшить размер кода и время исполнения.

Когда вы задаете `-O', компилятор включает `-fthread-jumps' и `-fdefer-pop' на всех машинах. компилятор включает `-fdelayed-branch' на машинах, которые имеют щели задержки, и `-fomit-frame-pointer' на машинах, которые могут обеспечивать отладку даже без указателя фрейма. На некоторых машинах компилятор также включает другие флаги.

-O2

Оптимизирует даже больше. GNU CC выполняет почти все поддерживаемые оптимизации, которые не включают уменьшение времени исполнения за счет увеличения длины кода. Компилятор не выполняет раскрутку циклов или подстановку функций, когда вы указываете `-O2'. По сравнения с `-O' эта опция увеличивает как время компиляции, так и эффективность сгенерированного кода.

`-O2' включает все необязательные оптимизации кроме раскрутки циклов и подстановки функций. Она также включает опцию `-fforce-mem' на всех машинах и исключение указателя фрейма на машинах, где это не помешает отладке.

-O3

Оптимизирует еще больше. `-O3' включает все оптимизации, определяемые `-O2', а также включает опцию `inline-functions'.

-O0

Не оптимизировать.

Если вы используете многочисленные `-O' опции с номерами или без номеров уровня, действительной является последняя такая опция.

Опции вида `-fфлаг' определяют машинно-независимые флаги. Большинство флагов имеют положительную и отрицательную формы; отрицательной формой `-ffoo' будет `-fno-foo'. В таблице, расположенной ниже, указана только одна из форм - та, которая не является умолчанием. Вы можете сконструировать другую форму удалением или добавлением '-no'.

-ffloat-store

Не позволяет сохранять переменные с плавающей точкой в регистрах и препятствует другим опциям, которые могут изменять, откуда берется значение с плавающей точкой - из регистра или из памяти.

Эта опция предотвращает нежелательную излишнюю точность на машинах типа 68000, где регистры с плавающей точкой (из 68881) хранят большую точность, чем имеет double. Для большинства программ, излишняя точность делает только лучше, но некоторые

программы полагаются на точное описание плавающей точки IEEE. Используйте `-ffloat-store' для таких программ.

-fno-defer-pop

Всегда выталкивает параметры каждого вызова функции сразу после возврата из нее. Для машин, которые должны выталкивать параметры после обращения к функции, компилятор обычно позволяет параметрам накапливаться на стеке при нескольких вызовах функций, и выталкивает их все сразу.

-fforce-mem

Заставляет копировать операнды в памяти в регистры перед выполнением с ними арифметических операций. Это производит лучший код, поскольку все ссылки становятся потенциальными общими подвыражениями. Если они не являются общими подвыражениями, комбинацирование команд должно исключить отдельную загрузку регистров. Опция '-O2' включает эту опцию.

-fforce-addr

Заставляет копировать константные адреса памяти в регистры перед выполнением с ними арифметических операций. Это производит лучший код точно также как и '-fforce-mem'.

-fomit-frame-pointer'

Не хранит указатель фрейма для функций, которые не нуждаются в нем. Это избегает команд сохранения, установки и восстановления указателя фрейма; это также делает доступным дополнительный регистр во многих функциях. Это делает отладку невозможной на некоторых машинах.

На некоторых машинах, таких как Vax, этот флажок не имеет эффекта, потому что стандартная вызывающая последовательность автоматически обрабатывает указатель фрейма, и ничего не экономится, если считается, что он не существует. Макрос `FRAME_POINTER_REQUIRED' в описании архитектуры управляет тем, обеспечивает ли целевая машина этот флаг или нет. См. раздел [Регистры].

-fno-inline

Не обращать внимание на ключевое слово inline. Обычно эта опция используется, чтобы уберечь компилятор от подстановки любых функций. Обратите внимание что, если вы не оптимизируете, никакие функции не могут быть подставлены.

-finline-functions

Интегрирует все простые функции в вызывающие функции. Компилятор эвристически решает, какие функции достаточно просты, чтобы их стоило интегрировать таким образом.

Если все вызовы к данной функции интегрированы, а функция обьявлена static, тогда ассемблерный код функции обычно не выводится в своем настоящем виде.

-fkeep-inline-functions

Даже если все вызовы к данной функции интегрированы, и функция обьявлена static, выводит, однако, отдельную вызываемую во время выполнения версию функции.

-fno-function-cse

Не кладет адреса функций в регистры; каждая инструкция, которая вызывает постоянную функцию явно содержит адрес функции.

Эта опция приводит к менее эффективному коду, но некоторые странные куски, которые изменяют ассемблерный код, могут быть спутаны оптимизацией, выполняемой, когда эта опция не используется.

-ffast-math

Эта опция позволяет GCC нарушать некоторые ANSI или IEEE правила и/или спецификации с целью оптимизации быстродействия кода. Наример, она позволяет компилятору считать, что параметры функции sqrt являются неотрицательными числами, и что никакие значения с плавающей точкой не являются NaNs.

Эта опция не должна никогда включаться любой из `-O' опций, так как она может приводить к неправильному результату для программ, которые зависят от точного

выполнения правил/спецификаций IEEE или ANSI для математических функций.

Следующие опции управляют специфическими оптимизациями. Опция `-O2' включает все эти оптимизации за исключением `-funroll-loops' и `-funroll-all-loops'. На большинстве машин опция '-O' включает опции `-fthread-jumps' и `-fdelayed-branch', но отдельные машины могут обрабатывать ее по-другому.

Вы можете использовать следующие флаги в тех редких случаях, когда желательно выполнять "тонкую настройку" оптимизации.

-fstrength-reduce

Выполняет оптимизацию циклов: понижение силы операций и исключение итеративных переменных.

-fthread-jumps

Выполняет оптимизацию условного перехода, если выясняется, что он переходит к участку, где находится другое сравнение подчиненное первому. Если так, то первый переход переназначается на точку назначения второго перехода или точку сразу после него, в зависимости от того, должно ли быть второе условие истинным или ложным.

-fcse-follow-jumps

В исключении общих подвыражений проходит через команду перехода, если точка назначения перехода не достигается никаким другим маршрутом. Например, когда CSE сталкивается с оператором if с ветвью else, CSE будет следовать переходу к случаю, когда проверяемое условие ложно.

-fcse-skip-blocks

Подобна `-fcse-follow-jumps', но заставляет CSE следовать переходам, которые условно пропускают блоки. Когда CSE сталкивается с простым оператором if без ветви else, `-fcse-skip-blocks' заставляет CSE следовать переходу вокруг тела if.

-frerun-cse-after-loop

Повторно запускается общее удаление подвыражений после оптимизации циклов.

-fexpensive-optimizations

Выполняет ряд малых оптимизации, которые являются относительно дорогими.

-fdelayed-branch

Если поддерживается для целевой машины, пытается переупорядочивать инструкции так, чтобы эксплуатировать щели задержки доступные после задержанной команды перехода.

-fschedule-insns

Если обеспечивается для целевой машины, пытается переупорядочить инструкции так, чтобы исключить простои исполнения из-за того, что отсутствуют требуемые данные. Это помогает машинам, которые имеют медленные инструкции с плавающей точкой или загрузки памяти, при разрешении другим командам выполняться, пока результат загрузки или инструкции с плавающей точкой не требуется.

-fschedule-insns2

Подобна `-fschedule-insns', но запрашивает дополнительный проход планировщика после того, как было выполнено распределение регистров. Это особенно полезно на машинах с относительно маленьким числом регистров и там, где команды загрузки памяти требуют больше чем один цикл.

-fcaller-saves

Дает возможность распредять значения в регистрах, которые затираются вызовами функций, с порождением дополнительных команд сохранения и восстановления регистров вокруг таких вызовов. Такое распределение выполняется только тогда, когда кажется, что оно приводит к лучшему коду чем другое.

Эта опция включается по умолчанию на некоторых машинах, обычно на тех, которые не имеют никаких сохраняемых при вызове регистров для использования взамен.

-funroll-loops

Выполняет раскрутку циклов. Она выполненяется только для циклов, число итераций которых можно определить в процессе компиляции или во время выполнения. `-funroll-

loop' подразумевает и `-fstrength-reduce' и `-frerun-cse-after-loop'.

-funroll-all-loops

Выполняет раскрутку циклов. Она выполняется для всех циклов и обычно делает программы более медленными. `-funroll-all-loops' подразумевает `-fstrength-reduce', также как и `-frerun-cse-after-loop'.

-fno-peephole

Отключает любые peephole-оптимизации специфические для машины.

2.7 Опции Управляющие Препроцессором

Эти опции управляют препроцессором C, который обрабатывает каждый C исходный файл перед фактической компиляцией.

Если вы используете опцию `-E', ничего кроме препроцессирования не делается. Некоторые из этих опций имеют смысл только вместе с `-E', потому что они делают вывод препроцессора неподходящим для фактической компиляции.

-include файл

Обрабатывает 'файл' как ввод перед обработкой обычного входного файла. Фактически, содержимое 'файла' компилируется сначала. Любая опция '-D' или '-U' из командной строки обрабатывается до '-include файл', вне зависимости от порядка, в котором они записаны. Все опции '-include' и '-imacros' обрабатываются в том порядке, в котором они записаны.

-imacros файл

Обрабатывает 'файл' как ввод, отбрасывая возникающий в результате вывод до обработки обычного входного файла. Так как вывод сгенерированный из 'файла' отбрасывается, единственый эффект `-imacros файл' состоит в том, что макрокоманды определенные в 'файле' становятся доступны для применения в главном вводе.

Любая опция '-D' или '-U' из командной строки обрабатывается до '-imacros файл', вне зависимости от порядка, в котором они записаны. Все опции '-include' и '-imacros' обрабатываются в том порядке, в котором они записаны.

-idirafter директорий

Добавляют каталог 'директорий' ко второму маршруту включения. В каталогах второго маршрута включения ищут, когда заголовочный файл не обнаружен ни в одном из каталогов в главном маршруте включения (маршрут, к которому добавляет опция `-I').

-iprefix префикс

Определяет 'префикс' как префикс для нижеследующей опции `-iwithprefix'.

-iwithprefix директорий

Добавляет каталог ко второму маршруту включения. Имя каталога получается объединением 'префикса' и 'директория', где 'префикс' определялся предварительно опцией `-iprefix'. Если вы еще не определили префикс, по умолчанию используется каталог, содержащий инсталированные проходы компилятора.

-iwithprefixbefore директорий

Добавляет каталог к главному маршруту включения. Имя каталога получается объединением 'префикса' и 'директория', как в случае `-iwithprefix'.

-isystem директорий

Добавляет каталог 'директорий' к началу второго маршрута включения, помечая его как системный каталог, так что он имеет ту же самую специальную обработку, что и стандартные системные каталоги.

-nostdinc

Не ищет в стандартных системных каталогах для заголовочных файлов. Только каталоги, которые вы определили опциями `-I' (и текущий каталог, если подходит) используется для поиска. См. раздел [Опции Ди- ректорий] для информации об '-I'.

Используя `-nostdinc' и `-I-', вы можете ограничить маршрут поиска файлов включения только теми каталогами, которые вы задали явно.

-undef

Не предопределяет любые нестандартные макросы. (Включая флаги архитектуры).

-E

Запускает только препроцессор C. Препроцессирует все указанные C исходные файлы и выводит результаты в стандартный вывод или в указанный выходной файл.

-C

Говорит препроцессору не отбрасывать комментарии. Используется с опцией `-E'.

-P

Говорит препроцессору не генерировать директивы `#line'. Используется с опцией `-E'.

-M

Говорит препроцессору выводить правила для make, описывающие зависимости каждого объектного файла. Для каждого исходного файла, препроцессор выводит одно make-правило, чья цель - имя объектного файла, для которого исходный файл и чьими зависимостями являются все #include заголовочные файлы, которые он использует. Это правило может быть одиночной строкой или может быть продолжено с помощью `\'-новая строка, если оно длинное. Список правил печатается в стандартный вывод вместо препроцессированной C программы.

`-M' подразумевает `-E'.

Другой способ указать вывод make-правил - с помощью установки переменной окружения DEPENDENCIES_OUTPUT (См. раздел [Переменные Окружения]).

-MM

Подобна `-M', но вывод упоминает только заголовочные файлы пользователя, включенные с помощью `#include "файл"'. Системные заголовочные файлы, включенные с помощью `#include <файл>' опускаются.

-MD

Подобна `-M', но информация о зависимостях записывается в файл получающемуся при замене ".c" на ".d" на концах имен входных файлов. Это делается в добавление к указанной компиляции файла - `-MD' не запрещает обычную компиляцию, как это делает '-M'.

В Mach вы можете использовать утилиту `md', чтобы обьединить многочисленные файлы зависимостей в один файл, подходящий для использования с командой make.

-MMD

Подобна `-MD', за исключением упоминания только пользовательских заголовочных файлов, но не системных заголовочных файлов.

-MG

Обрабатывать отсутствующие заголовочные файлы как генерируемые файлы и считать, что они находятся в том же самом каталоге, что и исходный файл. Если вы указываете `-MG', вы также должны указывать или `-M', или `-MM'. `-MG' не поддерживается с `-MD' или `-MMD'.

-H

Печатает имя каждого используемого заголовочного файла в добавление к нормальным действиям.

-Авопрос(ответ)

Утвердить ответ 'ответ' для 'вопроса', в случае, если он проверяется условным выражением препроцессора типа `#if #вопрос(ответ)'. `-A-' отключает стандартные утверждения, которые обычно описывают целевую машину.

-Dмакрос

Определяет макрос 'макрос' со строкой `1' в качестве его определения.

-Dмакрос=определение'

Определяет макрос 'макрос' как 'определение'. Все экземпляры `-D' в командной строке обрабатываются до любых опций `-U'.

-Uмакрос

Отменяют определение макроса 'макрос'. Опции `-U' обрабатываются после всех опций `-D', но перед любыми `-include' и `-imacros' опциями.

-dM

Говорит препроцессору вывести только список макроопределений, которые имеют действие в конце препроцессирования. Используется с опцией `-E'.

-dD

Говорит препроцессору передать все макроопределения в вывод в их последовательности в другом выводе.

-dN

Подобна `-dD', за исключением того, что макроаргументы и содержание опускаются. В вывод включается только `#define имя'.

-trigraphs

Поддерживает трехзнаковые последовательности ANSI C. Опция `-ansi' также имеет этот эффект.

-Wp,опция

Передает 'опцию' в качестве опции препроцессору. Если 'опция' содержит запятые, она расщепляется запятыми на многочисленные опции.

2.8 Передача Опций Ассемблеру

Вы можете передать опции ассемблеру.

-Wa,опция

Передает 'опцию' в качестве опции ассемблеру. Если 'опция' содержит запятые, она расщепляется запятыми на многочисленные опции.

2.9 Опции Линковки

Эти опции вступяют в игру тогда, когда компилятор линкует объектные файлы в выполнимый выходной файл. Они бессмысленны, если компилятор не выполняет шаг линковки.

имя-объектного-файла

Имя файла, которое не заканчивается специальным распознаным суффиксом, считается именем объектного файла или библиотеки. (Объектные файлы отличаются линкером от библиотек по содержимому файла.) Если выполняется линковка, эти объектные файлы используются в качестве ввода для линкера.

-c

-S

-E

Если используется любая из этих опций, то линкер не запускается, и имена объектных файлов не должны использоваться в качестве параметров. См. раздел [Общие Опции].

-lбиблиотека

Ищет при линковке библиотеку с именем 'библиотека'.

Есть различие в том, где в комадной строке вы записываете эту опцию; линкер ищет обрабатываемые библиотеки и объектные файлы в порядке, в котором они указаны. Таким образом, `foo.o -lz bar.o' ищет библиотеку `z' после файла `foo.o', но перед `bar.o'. Если `bar.o' ссылается на функции в `z', эти функции не могут быть загружены.

Линкер просматривает стандартный список каталогов в поиске библиотеки, который, фактически, является файлом с именем `libбиблиотека.a'. Затем линкер использует этот файл так, как будто бы он был точно специфицирован по имени.

Директории, в которых ищет линкер, включают несколько стандартных системных каталогов, плюс любые каталоги, которые вы определяете с помощью `-L'.

Обычно файлы, обнаруженные этим способом являются библиотечными файлами - архивными файлами, чьи элементы - объектные файлы. Линкер обрабатывает архивный файл, просматривая его в поиске элементов, которые определяют символы, на которые были ссылки, но которые до сих пор не определялись. Но, если оказывается, что обнаруженный файл - обычный объектный файл, то он линкуется в обычном порядке. Единственное различие между использованием опции `-l' и указанием имени файла в том, что `-l' добавляет к 'библиотеке' `lib' и `.a' и ищет в нескольких директориях.

-lobjc

Вам нужен этот специальный случай опции `-l' для линковки Objective C программ.

-nostartfiles

Не использует стандартные системные файлы начального запуска при линковке. Стандартные системные библиотеки используются как обычно, если не указано `-nostdlib' или `-nodefaultlibs'.

-nodefaultlibs

Не использует стандартные системные библиотеки при линковке. Линкеру передаются только те библиотеки, которые вы указываете. Стандартные файлы начального запуска используются как обычно, если не указано `-nostartfiles'.

-nostdlib

Не использует стандартные системные файлы начального запуска и библиотеки при линковке. Никакие файлы запуска и только те библиотеки, которые вы указываете будут, переданы линкеру.

Одной из стандартных библиотек, обходимых `-nostdlib' и `-nodefaultlibs' является `libgcc.a' - библиотека внутренних подпрограмм, которые использует GNU CC, чтобы преодолевать изъяны конкретных машин, или для специальных потребностей некоторых языков. (См. Главу 13 [Интерфейс к выводу GNU CC] для подробного рассмотрения `libgcc.a'.) В большинстве случаев, вы нуждаетесь в `libgcc.a', даже когда вы хотите избегать других стандартных библиотек. Другими словами, когда вы указываете `-nostdlib' или `-nodefaultlibs', вы должны обычно также указывать `-lgcc'. Это гарантирует, что нет никаких нарушеных ссылок к внутренним библиотечным подпрограммам GNU CC. (Например, `__main' , используемая, чтобы гарантировать вызов конструкторов C++.)

-s

Удаляет все символьные таблицы и информацию о перемещениях из исполняемого файла.

-static

В системах, которые поддерживают динамическую линковку, предотвращает линковку с разделяемыми библиотеками. В других системах, эта опция не имеет никакого эффекта.

-shared

Производит разделяемый объект, который может затем быть слинкован с другими объектами, чтобы сформироваться исполнимый файл. Только некоторые системы поддерживают эту опцию.

-symbolic

Связывает ссылки к глобальным символам при формировании разделяемого объекта. Предупреждает о любых неразрашенных ссылок (если не отменено опцией редактора связей `-Xlinker -z -Xlinker определения'). Только некоторые системы поддерживают эту опцию.

-Xlinker опция

Передает 'опцию' в качестве опции линкеру. Вы можете использовать ее, чтобы поддерживать специфические для системы опции линкера, о которых GNU CC не знает

как распознавать.

Если вы хотите передать опцию, которая имеет параметр, вы должны использовать `-Xlinker' дважды: один раз для опции и один раз для параметра. Например, чтобы передать `-assert описания', вы должны написать `-Xlinker -assert -Xlinker описания'. Не срабатывает, если написать `-Xlinker " -assert описания " ', потому что это передает всю строку как единый параметр, который не ожидается линкером.

-Wl,опция

Передает 'опцию' в качестве опции линкеру. Если 'опция' содержит запятые, она расщепляется запятыми на многочисленные опции.

-uсимвол

Делает вид, что символ 'символ' неопределен, вынуждая линкуемые библиотечные модули определять его. Вы можете использовать `-u' много разы с различными символами, чтобы вызвать загрузку дополнительных библиотечных модулей.

2.10 Опции для Поиска в Директориях

Эти опции определяют каталоги для поиска заголовочных файлов, библиотек и частей компилятора:

-Iдиректория

Добавляет каталог 'директория' в начало списка каталогов, используемых для поиска заголовочных файлов. Ее можно использовать для подмены системных заголовочных файлов, подставляя ваши собственные версии, поскольку эти директории просматриваются до директорий системных заголовочных файлов. Если вы используете более чем одну опцию '-I', директории просматриваются в порядке слева направо; стандартные системные директории идут после.

-I-

Любые каталоги, которые вы указываете с опциями `-I' до опции `-I-' просматриваются только в случае `#include "файл"'; они не просматриваются для

`#include <файл>'.

Если дополнительные каталоги указаны с опциями `-I' после опции `-I-', эти каталоги просматриваются для всех директив `#include'. (Обычно все `-I' директории используются таким способом.)

Кроме того, опция `-I-' запрещает использование текущего каталога (откуда пришел текущий входной файл) в качестве первого каталога для поиска `#include "файл"'. Нет никакого способа отменить этот эффект опции `-I-'. С помощью `-I.' вы можете указать поиск в каталоге, который был текущим, когда вызывался компилятор. Это не точно то же самое, что делает препроцессор по умолчанию, но часто подходит.

'-I-' не запрещает использование стандартных системных директориев для заголовочных файлов. Таким образом, '-I-' и '-nostdinc' являются независимыми.

-Bпрефикс

Эта опция указывает где искать выполнимые, библиотечные и заголовочные файлы, а также файлы данных для самого компилятора.

Управляющая программа компилятора запускает одну или больше программ 'cpp', 'cc1', 'as' и 'ld'. Он пытается использовать 'префикс' в качестве префикса для каждой программы, которую он запускает, как с так и без 'машина/версия/' (см. раздел [Целевые Опции]).

Для каждой запускаемой программы компилятор сначала пытается использовать префикс '-B', если указан. Если такое имя не найдено, или опция '-B' не указана, компилятор пытается использовать два стандартных префикса: '/usr/lib/gcc/' и '/usr/local/lib/gcc-lib/'. Если ни один из этих префиксов не дает результата, не модифицированное имя программы ищется с использованием директорий указанных в вашей переменной окружения 'PATH'.

Префикс '-B', который эффективно указывает имена директорий, также применяется к библиотекам в линкере, поскольку компилятор преобразует эти опции в опции '-L' для линкера. Они, также, применяются к заголовочным файлам в препроцессоре, поскольку компилятор преобразует эти опции в опции '-isystem' для препроцессора. В этом случае компилятор добавляет 'include' к префиксу.

Файл поддержки времени выполнения 'libgcc.a' может также искаться, если нужно, с использованием префикса '-B'. Если он не найден там, используются два стандартных префикса, указанных выше, и это все. Файл пропадает из линковки, если он не найден таким способом.

Другой способ указать префикс очень сходный с префиксом '-B' - использовать переменную окружения GCC_EXEC_PREFIX. См. раздел [Переменные Окружения].

2.11 Указание Целевой Машины и Версии Компилятора

По умолчанию, GNU CC компилирует код для той же самой машины, которую вы используете. Однако, он также может быть инсталлирован как кросс-компилятор, чтобы компилировать для какого-нибудь другого типа машин. В действительности, несколько различных конфигураций GNU CC, для различных целевых машин, могут быть установлены бок о бок. Затем, вы указываете, какую конфигурацию использовать с помощью опции '-b'.

Кроме того, более старые и более новые версии компилятора могут быть установлены бок о бок. Одина из них (вероятно, самая новая) будет являться умолчанием, но вы можете захотеть иногда использовать другую.

-b машина

Аргумент 'машина' указывает целевую машину для компиляции. Она полезна, когда вы установили GNU CC в качестве кросс-компилятора.

Значение, используемое в качастве аргумента 'машина' - то же, что было указано при конфигурировании GNU CC в качестве кросс-компилятора. Например, если кросс-компилятор конфигурировался с 'configure i386v', означающем компиляцию для 80386 с System V, тогда вы должны указать '-b i386v', чтобы запустить этот кросс-компилятор.

Когда вы не указываете '-b', это обычно означает компиляцию для того же самого типа машины, который вы используете.

-V версия

Аргумент 'версия' указывает, какую версию компилятора запускать. Она полезна, когда установлены многочисленные версии. Например, 'версия' может быть '2.0', означая запуск GNU CC версии 2.0.

Версией по умолчанию, когда вы не указываете '-V', является последняя версия GNU CC, которую вы установили.

Опции '-b' и '-V' в действительности работают с помощью управления частью имен файлов, используемых для выполнимых файлов и библиотек, используемых при компиляции. Данная версия компилятор для данной целевой машины обычно храниться в директории '/usr/local/lib/gcc-lib/машина/версия'.

Таким образом, можно установить действие опций '-b' и '-V', либо изменив имена этих директорий, либо добавив альтернативные имена (или символические ссылки). Если в деректории '/usr/local/lib/gcc-lib/' файл '80386' является символической ссылкой на файл 'i386v', тогда '-b 80386' становиться синонимом '-b i386v'.

С одной стороны, опции '-b' и '-V' не совсем переключают на другой компилятор: управляющая программа верхнего уровня gcc, которую вы первоначально вызвали, продолжает выполняться и вызывать другие исполнимые файлы (препроцессор, сам компилятор, ассемблер и линкер), которые и выполняют реальную работу. Однако, поскольку никакой реальной работы не выполняется в управляющей программе, обычно не важно, что используемая управляющая программа не является программой для указанной машины и версии.

Единственное место, в котором управляющая программа зависит от целевой машины, в разборе и обработке специальных машинозависимых опций. Однако, это контролируется файлом, который находится, вместе с другими выполнимыми файлами, в директории для указанной версии и целевой машины. В результате, одна установленная управляющая программа адаптируется к любым указанным целевой машине и версии компилятора.

Выполнимый файл управляющей программы, однако, управляет одной важной вещью: версией компилятора и целевой машиной по умолчанию. Следовательно, вы можете установить различные экземпляры управляющей программы, скомпилированные для разных целевых машин и версий, под различными именами.

Например, если управляющая программа для версии 2.0 установлена как ogcc, а для версии 2.1 - как gcc, тогда команда gcc будет, по умолчанию, использовать версию 2.1, а ogcc -

версию 2.0. Однако, вы можете выбрать любую версию с любой командой с помощью опции '-V'.

2.12 Модели и Конфигурации Машин

Ранее мы обсуждали стандартную опцию '-b', которая выбирает среди различных инсталлированных компиляторов для совершенно различных типов целевых машин, таких как Vax, 68000, 80386.

В добавление, каждый из этих типов машин может иметь свои собственные специальные опции, начинающиеся с '-m', для выбора между различными моделями и конфигурациями машин. Например, 68010 или 68020, плавающий сопроцессор или нет. Одна установленная версия компилятора может компилировать для любой модели и конфигурации, в соответствии с указанными опциями.

Некоторые конфигурации компилятора также поддерживают дополнительные специальные опции, обычно для совместимости с другими компиляторами для той же платформы.

Эти опции определяются с помощью макроса TARGET_SWITCHES в описании архитектуры. Умолчания для опций также определяются этим макросом, который позволяет вам менять умолчания.

Опции Intel 386

Эти '-m' опции определены для семейства i386 компьютеров:

-m486

-m386

Управляет тем, оптимизируется ли код для 486 вместо 386 или нет. Код сгенерированный для 486 будет работать на 386 и наоборот.

-mieee-fp

-mno-ieee-fp

Управляет тем, использует или нет компилятор IEEE сравнения с плавающей точкой. Они корректно обрабатывают случай, когда результат сравнения неопределен.

-msoft-float

Порождает выход, содержащий библиотечные вызовы для плавающей точки. Предупреждение: необходимая библиотека не является частью GNU CC. Обычно используются способности обыкновенных компиляторов C, но это не может быть сделано прямо в кросс-компиляции. Вы должны провести свое собственное мероприятие, чтобы обеспечить подходящюю библиотеку функций для кросс-компиляции.

На машинах, где функции, возвращающие значение с плавающей точкой, выдают результат на регистровом стеке 80387, некоторые инструкции с плавающей точкой могут порождаться даже при использовании '-msoft-float'.

-mno-fp-ret-in-387

Не использует регистру плавающего сопроцессора для возврата значений из функций.

Обычная конвенция вызовов возвращает значения типов float и double на регистрах плавающего сопроцессора, даже, если плавающего сопроцессора нет. Идея состоит в том, что операционная система должна эмулировать плавающий сопроцессор.

Опция '-mno-fp-ret-in-387' заставляет вместо этого возвращать такие значения на обычных регистрах центрального процессора.

-mno-fancy-math-387

Некоторые эмуляторы 387 не поддерживают инструкции sin, cos и sqrt для 387. Указывая эту опцию, вы избегаете генерации этих инструкций. Эта опция включается по умолчанию на FreeBSD. В качестве исправления 2.6.1, эти инструкции не генерируются, если вы не используете, также, опцию '-ffast-math'.

-malign-double

-mno-align-double

Управляет тем, выравнивает ли GNU CC double, long double и long long переменные на границу двух слов или на границу одного слова. Выравнивание double переменных на границу двух слов порождает код, который выполняется немного быстрее на 'Pentium' при затрате большего количества памяти.

Предупреждение: Если вы используете опцию '-malign-double', структуры, содержащие выше перечисленные типы будут выравниваться не так, как в опубликованных спецификациях прикладного двоичного интерфейса для 386.

-msvr3-shlib

-mno-svr3-shlib

Управляет тем, куда GNU CC кладет неинициализированные локальные переменные - в bss или data. 'msvr3-shlib' кладет эти переменные в bss. Эти опции имеют смысл только в System V Release 3.

-mno-wide-multiply

-mwide-multiply

Управляет тем, использует ли GNU CC mul и imul, которые выдают 64-битный результат в eax:edx из 32-битных операндов для выполнения long long умножения и 32-битного деления на константы.

-mrtd

Использует другую конвенцию вызова функций, в которой функция, которая имеет фиксированное число аргументов возвращается с помощью инструкции 'ret число', которая выталкивает аргументы при возврате. Это сохраняет одну инструкцию в месте вызова, поскольку не надо выталкивать аргументы там.

Вы можете указать, что отдельная функция вызывается с этой конвенцией вызова с помощью атрибута функции 'stdcall'. Вы можете, также, отменить опцию '-mrtd', используя атрибут функции 'cdecl'. См. раздел 6.22 [Атрибуты Функций].

Предупреждение: эта конвенция вызова несовместима с обычно используемой в Unix, так что вы не можете использовать ее, если вам нужно вызывать библиотеки

скомпилированные компилятором Unix.

Вы, также, должны обеспечить прототипы для всех функций, которые имеют переменное число аргументов (включая printf); иначе для вызова этих функций будет генерироваться неверный код.

Кроме того, очень некорректный код будет сгенерирован, если вы вызовите функцию со слишком большим числом аргументов. (Обычно, лишние аргументы без последствий игнорируются.)

-mreg-alloc=регистры

Управляет порядком распределения по умолчанию целых регистров. Строка 'регистры' представляет собой ряд букв определяющих регистры. Поддерживаемыми буквами являются: a выделяет EAX, b выделяет EBX, c выделяет ECX, d выделяет EDX, S выделяет ESI, D выделяет EDI, B выделяет EBP.

-mreg-parm=число

Управляет тем, сколько регистров используется для передачи целых аргументов. По умолчанию, регистры для передачи параметров не используются, и максимум 3 регистра могут быть использованы. Вы можете управлять этим для конкретных функций используя атрибут функции 'regparm'. См. раздел 6.22 [Атрибуты Функций].

Предупреждение: если вы используете эту опцию, и 'число' не равно нулю, вы должны строить все модули с одним и тем же значением, включая все библиотеки. Это включает системные библиотеки и модули начальной загрузки.

-malign-loops=число

Выравнивает циклы на границу степень двух с показателем 'число' байт. Если '-malign-loops' не указана, умолчанием является 2.

-malign-jumps=число

Выравнивает инструкции, на которые только переходят на границу степень двух с показателем 'число' байт. Если '-malign-jumps' не указана, умолчанием является 2 при оптимизации для 386, и 4 - для 486.

-malign-functions=число

Выравнивает начала функций на границу степень двух с показателем 'число' байт. Если '-malign-functions' не указана, умолчанием является 2 при оптимизации для 386, и 4 - для 486.

2.13 Опции Соглашений о Генерации Кода

Эти машинонезависимые опции управляют соглашениями об интерфейсе, используемыми при генерации кода.

Большинство из них имеет как положительную, так и отрицательную формы; отрицательной формой '-ffoo' будет '-fno-foo'. В таблице ниже, указывается только одна из этих форм - та, которая не является умолчанием. Вы можете получить другую форму, либо удалив 'no-', либо добавив его.

-fpcc-struct-return

Возвращает "короткие" структуры и объединения в памяти, также как и длинные, а не на регистрах. Эта конвенция менее эффективна, но она имеет преимущество в совместимости вызовов между файлами, скомпилированными GNU CC, и файлами, скомпилированными другими компиляторами.

Точная конвенция возврата структур в памяти зависит от макросов описания архитектуры.

Короткими структурами и объединениями являются те, чей размер и выравнивание соответствует одному из целых типов.

-freg-struct-return

Использует соглашение, что структуры и объединения возвращаются, когда возможно. на регистрах. Это более эффективно для малых структур, чем '-fpcc-struct-return'.

Если вы не указываете ни '-fpcc-struct-return', ни ее противоположность - '-freg-struct-return', GNU CC использует стандартную для данной архитектуры конвенцию. Если стандартной конвенции не существует, GNU CC использует '-fpcc-struct-return',

кроме архитектур, где GNU CC является основным компилятором. В этом случае мы можем выбрать стандарт, и мы выбираем более эффективный вариант регистрового возврата.

-fshort-enums

Выделяет для типа перечисления только такое количество байтов, которое нужно для объявленного диапазона возможных значений. А именно, тип перечисления будет эквивалентен наименьшему целому типу, который имеет достаточно места.

-fshort-double

Использует тот же размер для double, что и для float.

-fshared-data

Требует, чтобы данные и неконстантные переменные этой единицы компиляции были разделяемыми данными, а не личными. Это различие имеет смысл только для некоторых операционных систем, где разделяемые данные разделяются между процессами, выполняющимися в одной программе, в то время как личные данные существуют в одной копии на процесс.

-fno-common

Размешает даже неинициализированные глобальные переменные в секции bss объектного файла, вместо генерации их в качестве общих блоков.Это имеет эффект в том, что если одна и та же переменныя объявлена (без extern) в двух различных единицах компиляции, вы получети ошибку при их линковке. Единственной причиной, по которой это может быть полезно, является проверка того, что программа будет работать на других системах, которые всегда работают таким образом.

-fno-ident

Игнорирует директиву '#ident'.

-finhibit-size-directive

Не выводит ассемблерную директиву '.size', или что-нибудь еще, что может вызвать проблемы, если функция расщепляется посередине, и две половинки размещаются в областях, далеко отстоящих друг от друга в памяти. Эта опция используется при компиляции 'crtstuff.c'; вам не должно понадобиться использовать ее для чего-нибудь еще.

-fverobose-asm

Помещает дополнительную комментирующую информацию в ассемблерный код, чтобы сделать его более читаемым. Эта опция обычно нужна только тем, кому действительно нужно читать генерируемый ассемблерный код (может быть при отлаживании самого компилятора).

-fvolatile

Считает все ссылки в память через указатели volatile.

-fvolatile-global

Считает все ссылки в память на внешние и глобальные элементы данных volatile.

-fpic

Порождает позиционно-независимый код (position-independent code - PIC), подходящий для использования в разделяемой библиотеке, если поддерживается для целевой машины. Такой код берет все константные адреса из глобальной таблицы смещений (global offset table - GOT). Если размер GOT для линкуемого выполняемого файла превышает специфичный для машины максимальный размер, вы получаете сообщение об ошибке от линкера, показывающее, что '-fpic' не работает; в этом случае перекомпилируйте с заменой на '-fPIC'. (Эти максимумы составляют: 16K на m88k, 8K на Sparc и 32K на m68k и RS/6000. 386 не имеет такого лимита.)

Позиционно-независимый код требует специальной поддержки, и, следовательно, работает толька на некоторых машинах. Для 386, GNU CC поддерживает PIC для System V, но не для Sun 386i. Код генерируемый для IBM RS/6000 всегда позиционно-независимый.

Ассемблер GNU не полностью поддерживает PIC. На данный момент, вы должны использовать какой-нибудь другой ассемблер, чтобы PIC работал. Мы бы приветствовали добровольцев, которые улучшили бы GAS, чтобы обрабатывать PIC; первой частью работы является изложить, что ассемблер должен делать иначе.

-fPIC

Если поддерживается для целевой машины, порождает позиционно независимый код, подходящий для динамической линковки и не имеющий никаких ограничений на размер глобальной таблицы смещений. Эта опция дает отличие на m68k, m88k и Sparc.

Позиционно-независимый код требует специальной поддержки, и, следовательно, работает толька на некоторых машинах.

-ffixed-регистр

Обращается с регистром с именем 'регистр' как с фиксированным регистром; порождаемый код никогда не должен ссылаться на него (кроме, может быть, в качестве указателя стека, указателя фрейма или в какой-нибудь другой фиксированной роли).

'регистр' должно быть именем регистра. Допускаемые имена регистров являются машинозависимыми и определяются макросом REGISTER_NAMES в заголовочном файле описания архитектуры.

Эта опция не имеет отрицательной формы, потому что она определяет выбор из трех альтернатив.

-fcall-used-reg

Обращается с регистром с именем 'регистр' как с регистром, подходящим для распределения, который затирается вызовами функций. Он может выделяться для временных переменных или для переменных, которые не переживают вызов функции. Функции, откомпилированные с этой опцией, не будут сохранять и восстанавливать регистр 'регистр'.

'регистр' должно быть именем регистра. Допускаемые имена регистров являются машинозависимыми и определяются макросом REGISTER_NAMES в заголовочном файле

описания архитектуры.

Эта опция не имеет отрицательной формы, потому что она определяет выбор из трех альтернатив.

-fcall-saved-регистр

Обращается с регистром с именем 'регистр' как с регистром, подходящим для распределения, который сохраняется функциями. Он может выделяться даже для временных переменных или обычных переменных, которые переживают вызов функции. Функции, откомпилированные с этой опцией, будут сохранять и восстанавливать регистр 'регистр', если они его используют.

Использование этого флага для регистра, который имеет фиксированную роль в машинной модели исполнения такую, как указатель стека или указатель фрейма, порождает разрушительный результат.

Различные бедствия могут произойти от использования этого флага для регистров, в которых могут возвращаться значения функций.

Эта опция не имеет отрицательной формы, потому что она определяет выбор из трех альтернатив.

-fpack-struct

Упаковывает все члены структур рядом без зазоров. Обычно, не хотелось бы использовать эту опцию, поскольку она делает код не оптимальным, а смещения членов структур несоответствующим системным библиотекам.

2.14 Переменные Окружения, Затрагивающие GNU CC

Этот раздел описывает несколько переменных окружения, которые затрагивают то как работает GNU CC. Они работают указывая директории и префиксы для использования при поиске различных типов файлов.

Заметим, что вы также можете указать места для поиска, используя опции, такие как '-B', '-I' и '-L' (см. раздел [Опции Директорий]). Они имеют приоритет перед местами,

указанными с помощью переменных ок- ружения, которые, в свою очередь, имеют приоритет перед местами, ука- занными конфигурацией GNU CC. См. раздел 17.1 [Управляющая Программа].

TMPDIR

Если TMPDIR установлена, она указывает директорию, используемую для временных файлов. GNU CC использует временные файлы чтобы держать выход одной стадии компиляции, которая должна использоваться в качестве входа для следующей стадии: например, выход препроцессора, который является входом для собственно компилятора.

GCC_EXEC_PREFIX

Если GCC_EXEC_PREFIX установлена, она указывает префикс, используемый в именах программ, выполняемых компилятором. Косая черта не добавляется, когда этот префикс объединяется с именем программы, но вы можете указать префикс, который оканчивается косой чертой.

Если GNU CC не может найти программу, используя указанный префикс, он пытается смотреть в обычные места для программ.

Значением по умолчанию для GCC_EXEC_PREFIX является 'префикс/lib/gcc-lib/', где 'префикс' - значение prefix, когда вы запускали 'configure'.

Другие префиксы, указанные с помощью '-B', имеют приоритет перед этим префиксом.

Этот префикс также используется для нахождения файлов, таких как 'crt0.o', который используется для линковки.

Кроме того этот префикс используется необычным способом для нахождения директорий для поиска заголовочных файлов. Для каждой из стандартных директорий, чье имя обычно начинается с '/usr/local/lib/gcc-lib' (более точно, с значения GCC_INCLUDE_DIR), GNU CC пытается заменить это начало на указанный префикс, чтобы породить имя альтернативной директории. Таким образом, с '-Bfoo/' GNU CC будет искать 'foo/bar', если обычно он искал бы 'usr/local/lib/bar'. Эта альтернативная директория просматривается первой, затем идут стандартные директории.

COMPILER_PATH

Значением COMPILER_PATH является список директорий, разделенных двоеточиями, во многом схожий с PATH. GNU CC просматривает директории, указанные таким образом, при поиске программ, если он не может найти программы используя GCC_EXEC_PREFIX.

LIBRARY_PATH

Значением LIBRARY_PATH является список директорий, разделенных двоеточиями, во многом схожий с PATH. Когда GNU CC сконфигурирован как родной компилятор, он просматривает директории, указанные таким образом, при поиске специальных файлов линкера, если он не может найти их используя GCC_EXEC_PREFIX. Линковка при использовании GNU CC также использует эти директории при поиске обычных библиотек для опции '-l' (но директории, указанные с '-L', идут первыми).

C_INCLUDE_PATH

CPLUS_INCLUDE_PATH

OBJC_INCLUDE_PATH

Эти переменные окружения относятся к конкретным языкам. Значением каждой переменной является список директорий, разделенных двоеточиями, во многом схожий с PATH. Когда GNU CC ищет заголовочные файлы, он просматривает директории, перечисленные в переменной для языка, который вы используете, после директорий указанных с помощью '-I', но до стандартных директорий заголовочных файлов.

DEPENDENCIES_OUTPUT

Если эта переменная установлена, ее значение указывает как выводить зависимости для Make, основанные на заголовочных файлах, обработанных компилятором. Этот вывод во многом похож на вывод опции '-M' (см. раздел [Опции Препроцессора]), но он идет в отдельный файл и является добавлением к обычным результатам компиляции.

Значение DEPENDENCIES_OUTPUT может содержать только имя файла, в этом случае Make-правила пишутся в этот файл, а целевое имя извлекается из имени исходного файла.

Или же, значение может иметь форму 'файл цель', в этом случае правила пишутся в файл 'файл' с использованием 'цели' в качестве целевого имени.

2.15 Выполнение Protoize

Программа protoize является необязательной частью GNU C. Вы можете использовать ее, чтобы добавить прототипы к программе, конвертируя, таким образом, программу к ANSI C в одном отношении. Сопутствующая программа unprotioze делает обратное: она удаляет типы аргументов из всех прототипов, которые она находит.

Когда вы запускете эту программу, вы должны указать набор исходных файлов в качестве аргументов командной строки. Программа конверсии начинает работу с того, что компилирует эти файлы, чтобы узнать, какие функции в них определены. Информация, собранная о файле foo, сохраняется в файле с именем 'foo.X'.

После сканирования идет реальное преобразование. Указанные файлы все могут быть преобразованы; любые файлы, которые они включают (исходные или только заголовочные) также могут быть преобразованы.

Но не все подходящие файлы преобразуются. По умолчанию, protoize и unprotoize преобразуют только исходные файлы и файлы заголовка в текущем каталоге. Вы можете указать дополнительные каталоги, файлы в которых должны преобразовываться, с помощью опции `-dдиректория'. Вы можете также указать отдельные исключаемые файлы с помощью опции '-хфайл'. Файл преобразуется, если он подходит, имя его каталога соответствует одному из указанных имен каталогов, и его имя в каталоге не исключалось.

Основное преобразование protoize состоит в переписывании большинства описаний функций и обьявлений функций, чтобы указывать типы параметров. Не перезаписываются только объявления и описания функций с переменным числом аргументов.

protoize по выбору вставляет обьявления прототипов в начало исходного файла, делая их доступными для любых вызовов, которые предшествуют описанию функции. Или она может вставлять обьявления прототипов с блочной областью действия в блоках, где вызывается необъявленная функция.

Основное преобразование unprotoize состоит в переписывании большинства обьявлений функций так, чтобы удалить типы параметров, и в приведении описаний функций к форме старого pre-ANSI стиля.

Обе программы преобразования печатают предупреждение для любого обьявления или описания функции, которое они не могут преобразовать. Вы можете подавить эти предупреждения с помощью `-q'.

Вывод protoize или unprotoize заменяет первоначальный исходный файл. Первоначальный файл переименовывается к имени, оканчивающемуся на `.save'. Если `.save' файл уже существует, исходный файл просто отбрасывается.

И protoize, и unprotoize требуется сам GNU CC, чтобы просматривать программу и собирать информацию о функциях, которые она использует. Так что, ни одна из этих программ не будет работать до инсталяции GNU CC.

Ниже расположена таблица опций, которые вы можете использовать с protoize и unprotoize. Каждая опция работает с обеими программами, если не сказано иначе.

-Вдиректория

Ищет файл `SYSCALLS.c.X' в 'директории', вместо обычного каталога (обычно `/usr/local/lib'). Этот файл содержит информацию информацию о прототипах стандартных системных функций. Эта опция применима только к protoize.

-сопции-компиляции

Использует 'опции-компиляции' в качестве опций при запуске gcc для порождения `.X' файлов. Специальная опция `-aux-info' всегда передается в добавление, говоря gcc записать `.X' файл.

Обратите внимание, что опции компиляции нужно передавать как одиночный параметр protoize или unprotoize. Если вы хотите указать несколько опций gcc, вы должны заключить весь набор опций трансляции в кавычки, чтобы сделать их одним словом для shell'a.

Есть определенные параметры gcc, которые вы не можете использовать, потому что они производят неправильный вид вывода. Они включают `-g', `-O', `-c', `-S' и `-o'. Если вы включаете их в 'опции-компиляции', они игнорируются.

-C

Переименовывает файлы так, чтобы они заканчивались на `.C' вместо `.c'. Это удобно, если вы преобразуете C программу в C++. Эта опция применима только к protoize.

-g

Добавляет явные глобальные обьявления. Это значит вставлять явные обьявления в начало каждого исходного файла для каждой функции, которая вызывается в нем, и не была обьявлена. Эти обьявления предшествуют первому описанию функции, которое содержит вызов к необъявленной функции. Эта опция применима только к protoize.

-iстрока

Выравнивает обьявления параметров в старом стиле с помощью строки 'строка'. Эта опция применима только к unprotoize.

unprotoize преобразовывает описание-прототип функции в описание функции старого стиля, где параметры обьявлены между списоком параметров и начальным `{'. По умолчанию, unprotoize использует пять пробелов в качестве выравнивания. Если вы хотите вместо этого выравнивать только с одним пробелом, используйте `-i " "'.

-k

Сохраняет `.X' файлы. Обычно, они удаляются после завершения преобразования.

-l

Добавляет явные локальные обьявления. protoize с опцией `-l' вставляет обьявления-прототипы для каждой функции в каждом блоке, который вызывает функцию без обьявления. Эта опция применима только к protoize.

-n

Не делает никаких реальных изменений. Этот режим только печатает информацию о преобразованиях, которые были бы выполнены без `-n'.

-N

Не делает `.save' файлов. Первоначальные файлы просто удаляются. Используйте эту опцию с осторожностью.

-p программа

Использует программу 'программа' как компилятор. Нормально, используется имя `gcc'.

-q

Работает молча. Большинство предупреждений подавляется.

-v

Печатает номер версии, точно так же как `-v' для gcc.

Если вам нужны специальные опции компилятора, чтобы компилировать один из исходных файлов вашей программы, тогда вы должны сгенерировать соответствующий `.X' файл особенно, при выполнении gcc на этом исходном файле с соответствующими опциями и опцией `-aux-info'. Затем выполните protoize на всем наборе файлов. protoize будет использовать существующий `.X' файл, потому что он более новый, чем исходный файл. Например:

```
gcc -Dfoo=bar file1.c -aux-info
protoize * .c
```

Вы должны включать специальные файлы наряду с остальными в команде protoize, даже если их `.X' файлы уже существуют, потому что иначе они не будут преобразовываться.

См. раздел [Предостережения Protoize], для большей информации о том, как успешно использовать prtoize.

Вперед Назад Содержание