cc(1)

НАЗВАНИЕ

cc - компилятор C

СИНТАКСИС

```
cc [опции] файл ...
```

ОПИСАНИЕ

Команда **cc** - это интерфейс к системе компиляции языка C. Система компиляции состоит из следующих концептуальных стадий: *препроцессирование*, *компиляция*, *оптимизация*, *базовое профилирование* блоков, *ассемблирование* и *компоновка*. Команда **cc** проверяет введенные опции и суффиксы имен файлов, а затем выполняет необходимые стадии с соответствующими опциями и аргументами.

Команда **cc** распознает следующие суффиксы имен файлов:

.c	Файл, содержащий необработанный исходный код на языке C; применяются все стадии.
.i	Файл, содержащий обработанный препроцессором код на языке C; применяются все стадии, кроме препроцессирования.
.s	Файл, содержащий исходный текст на языке ассемблера; применяются только стадии ассемблирования и компоновки.
другой	Файл, содержащий данные, применимые только на стадии компоновки. В эту категорию обычно попадают объектные файлы (**.o**), архивные библиотеки (**.a**) и *совместно используемые* (*разделяемые*) объектные библиотеки (**.so**).

Если опции не указаны, команда **cc** проводит все файлы через все стадии обработки (в соответствии с их суффиксами), необходимые для получения на их основе динамически скомпонованного выполняемого файла **a.out** в текущем каталоге. Если запрошены промежуточные результаты, они тоже размещаются в текущем каталоге в файлах с именами, получаемыми путем замены входного суффикса на суффикс, соответствующий результирующему файлу. Если проходится стадия ассемблирования, команда **cc** помещает объектный файл (**.o**) в текущем каталоге, но этот файл будет удален, если единственный исходный файл компилируется в выполняемый. Все остальные промежуточные файлы помещаются во временный каталог. (Можно управлять используемым для этого каталогом с помощью переменной среды **TMPDIR**.)

Имеется исключение из правила суффиксов имен файлов, состоящее в том, что при указании опции **-E** будет воспринято любое имя файла, независимо от суффикса. Это позволяет вызывать препроцессор C как отдельное средство для применения к файлам, не содержащим исходный текст на языке C.

Следующие опции применимы ко всем стадиям.

-Q строка	Управляет включением идентификационной информации о средстве компиляции в результат. Если в качестве **строки** указан **y** (по умолчанию), эта информация будет включена; если же **n**, - не будет.
-V	Вызывает выдачу в стандартный поток ошибок информации о версии команды **cc** и каждого вызываемого средства, построчно.

-W стадия,список Передает аргумент(ы), задаваемые в **списке** через запятую (в указанном порядке) одной стадии компиляции, задаваемой параметром **стадия**. Аргумент в списке может содержать запятую, если она замаскирована обратной косой (\). Если несколько концептуальных стадий реализуются одним инструментальным средством, все соответствующие списки аргументов передаются этому средству. Стадии указываются следующим образом:

- **p** препроцессор
- **0** компилятор
- **2** оптимизатор
- **b** базовый профилировщик блоков
- **a** ассемблер
- **l** компоновщик

По отношению к обычным опциям и аргументам, переданным стадии компиляции, расположение аргументов из **списка** не определено и может изменяться.

-X строка Управляет уровнем соответствия стандартам ANSI и ISO для языка C. Аргумент опции **строка** может иметь одно из следующих значений:

- **a** Задает стандартное соответствие, за исключением того, что некоторые предупреждения не выдаются, а пространство имени расширено и включает имена, не задаваемые стандартами. Все конструкции C работают так, как указано в стандартах. Доступны также все реализованные расширения языка и библиотек, выходящие за пределы стандартов. Этот уровень соответствия используется по умолчанию.
- **c** Задает строгое соответствие стандартам. Поскольку пространство имен языка и заголовочных файлов сокращено по отношению к **-Xa**, определенные расширения (такие как ключевое слово **asm**) и некоторые общеупотребительные объявления заголовочных файлов недоступны. Последнюю проблему можно частично решить, используя **-D_POSIX_SOURCE** или **-D_XOPEN_SOURCE**.
- **t** Задает соответствие стандартам, за исключением семантики, отличающейся от "классического" C. (См. Kernighan & Ritchie, Первое издание.) Кроме того, выдаются предупреждения об изменениях семантики на стадии препроцессирования, таких как новые управляющие последовательности (вроде **\x**) и замены любых триграфов (вроде **??!**). Некоторые потенциальные возможности оптимизации, доступные в других режимах **-X**, также отключены.

Во всех режимах **-X**, стадия компиляции выдает предупреждения о выражениях, в которых измененные правила расширения более узкого диапазона значений без знака могут привести к скрытым (если не выдавать предупреждения) изменениям в работе.

-Y строка,каталог Использует указанный **каталог** для поиска элемента (элементов), указанных в **строке**. Аргумент **строка** может иметь одно или несколько из следующих значений:

- **стадия** Заставляет искать программу для указанной **стадии** компиляции (которая задается так же, как и для **-W**) в указанном **каталоге**. Если несколько концептуальных стадий реализуются одним инструментальным средством и для этих стадий указаны различные каталоги, выбор используемого каталога не определен.
- **I** Изменяет каталог, просматриваемый последним в поисках заголовочных файлов на стадии препроцессирования.
- **P** Изменяет стандартный путь поиска библиотек на стадии компоновки на список каталогов через двоеточие, задаваемый параметром **каталог**. (В начало этого пути за счет других опций могут быть добавлены другие каталоги.)
- **S** Изменяет каталог, из которого берутся зависящие от реализации *стартовые* (start-up) объектные файлы.

В дополнение к возможности задавать каталоги для поиска средств выполнения различных стадий компиляции, если команда **cc** вызвана как ***префикс*cc**, то каждое используемое средство будет предварено таким же префиксом, как и имена стартовых объектных файлов. Например, если команда **./abccc** вызвана с опцией **-Ya,../xyz**, предполагается, что ассемблер находится в файле **../xyz/abcas**.

Следующие опции применимы только на стадии препроцессирования. В ходе этой стадии всегда предопределены макросы **__STDC__** и **__USLC__**. Тогда как **__USLC__** всегда имеет положительное целое значение (которое означает, что используется компилятор USL C), **__STDC__** имеет значение 1 только для **-Xc**, в противном случае его значение - 0.

-A имя[(лексемы)] Вызывает проверку **имени** как предиката с необязательными лексемами в круглых скобках, как если бы он был указан в директиве **#assert**.

-A - Делает неопределенными все предопределенные макросы (кроме начинающихся на __) и отменяет проверку всех утверждений **#assert**.

Опция влияет на следующие предопределенные макросы и проверяемые утверждения:

```
#define i386 1        /* нет, если указана
                          опция -Xc */
#define unix 1        /* нет, если указана
                          опция -Xc */
#assert system(unix)
#assert cpu(i386)
#assert machine(i386)
```

-C Оставляет все комментарии в строках, не содержащих директивы, в результате препроцессирования; в противном случае они удаляются.

-D имя[=лексемы] Определяет **имя** как макрос, заменяемый на **лексемы** или на **1**, если конструкция **=лексемы** не указана, как если бы он был определен директивой **#define**.

-E Выполняет только стадию препроцессирования и посылает результат в стандартный выходной поток. Результат будет содержать строки, подобные директивам препроцессора, и может использоваться для последующей стадии компиляции. Имя обрабатываемого препроцессором файла может иметь любой суффикс.

-H Вызывает выдачу в стандартный поток ошибок полных имен всех включаемых файлов, по одному в строке.

-I каталог Вызывает просмотр указанного **каталога** в поисках включаемых файлов, имена которых не начинаются с /, перед просмотром стандартных каталогов. Если используется несколько опций **-I**, каталоги просматриваются в указанном порядке.

-P Выполняет только стадию препроцессирования и помещает результат в файл с суффиксом **.i**. В отличие от **-E**, результат не будет содержать директив. Компиляция получившегося файла **.i** даст в результате выполняемый файл, номера строк в котором будут соответствовать файлу **.i**, а не файлу **.c**.

-U имя Вызывает отмену определения **имени** как макроса, как если бы применялась директива **#undef**, даже если **имя** является предопределенным макросом (включая начинающиеся с __) или определено опцией **-D**.

Следующие опции применимы ко всем стадиям, кроме стадии препроцессирования. Все опции, влияющие только на стадию компоновки, передаются также команде **ld(1)**.

-B строка Управляет механизмом поиска имен библиотек на стадии компоновки для последующих опций **-l**. Аргумент опции, **строка**, может быть либо **dynamic** (начальная установка), либо **static**. Порядок указания опций **-B**, **-L** и **-l** имеет значение; см. описание опции **-l**.

-c Пропускает фазу компоновки. Созданные объектные файлы (**.o**) не удаляются.

-d строка Определяет режим создания выполняемого файла при компоновке. Если **строка** - это **y** (по умолчанию), будет создаваться динамически скомпонованный выполняемый файл; если **строка** - **n**, результат будет скомпонован статически.

-G Ведет к созданию компоновщиком *разделяемого* (shared object, *динамически подключаемая библиотека*), вместо выполняемого файла.

-g Вызывает генерацию информации, помогающей при символьной отладке. Эта опция конфликтует с **-O**, но имеет более низкий приоритет: если указаны обе опции, отладочная информация не генерируется.

-K список Включает различные варианты генерации кода, оптимизации или компоновки, или их комбинации. Для элементов следующего списка, представленных группами по два и более, первый элемент в такой группе определяет стандартное значение, причем действовать будет ровно один элемент группы. Аргумент опции **список** - это список одного или нескольких следующих элементов через запятую:

PIC	Изменяет генерацию кода так, что она не зависит от позиции.
thread	Указывает, будет ли программа использовать средства многопоточной обработки. В режиме с поддержкой многопоточности должны быть включены соответствующие флаги препроцессора и должна компоноваться в нужном месте *библиотека многопоточной обработки* (threading library).
blended **pentium** **i486** **i386**	Включает генерацию кода, специально оптимизированную для указанного процессора. **blended** балансирует варианты генерации кода так, чтобы они хорошо работали на всех процессорах. Используется только совместно с опцией **-O**.
ieee **no_ieee**	Управляет тем, будет ли генерируемый код вычислений с плавающей точкой строго соответствовать стандартам IEEE и C. **ieee** задает строгое соответствие. **no_ieee** разрешает более агрессивную оптимизацию. При этом предполагается, что приложение не изменяет режимов округления, не проверяет возникновение исключительных ситуаций и не генерирует пренебрежимо малых или бесконечно больших величин. В этом режиме **errno** может устанавливаться реже, а исключительные ситуации могут не возбуждаться.
no_frame **frame**	Определяет, должен ли генерируемый код использовать регистр **%ebp** как *указатель на фрейм стека* (stack frame pointer). **no_frame** позволяет использовать **%ebp** как регистр общего назначения, что может сделать образы стека при отладке нечитаемыми, но обычно приводит к более быстрому коду. Применим только, если указана опция **-O**. Неоптимизированный код всегда использует этот регистр как указатель фрейма.
host **no_host**	В режиме **host** компилятор предполагает, что имена стандартных функций C зарезервированы и эти функции работают так, как описано в стандартах. Компилятор может свободно *подставлять* (inline) такие функции вместо вызовов при необходимости. Если указан также **no_ieee**, компилятор предполагает, что математические функции не получают в качестве аргументов и не генерируют пренебрежимо малых или бесконечно больших величин.
no_inline **inline**	Определяет, должен ли компилятор выполнять подстановку функций. Подстановка функций может дать большую скорость выполнения за счет увеличения размера кода. Применим только, если указана опция **-O**.
loop_unroll **no_loop_unroll**	Определяет, должен ли компилятор выполнять *развертку цикла* (loop unrolling) при включенной оптимизации (**-O**). Развертка цикла может дать большую скорость выполнения за счет увеличения размера кода.
schar **uchar**	Определяет, должны ли символьные типы рассматриваться как знаковые или без знака. По умолчанию, эти типы рассматриваются как знаковые.

Несколько опций **-K** дают тот же эффект, как если бы отдельные списки аргументов были объединены в одной опции.

-L каталог Добавляет **каталог** на стадии компоновки к списку каталогов для просмотра последующими опциями **-l**. Порядок указания опций **-B**, **-L** и **-l** имеет значение; см. описание опции **-l**.

-l строка Заставляет компоновщик искать библиотеку **libстрока.so** или **libстрока.a**. Порядок указания опций **-B**, **-L** и **-l** имеет значение: опция **-l** заставляет компоновщик (в соответствующем порядке) проверять каталоги, указанные предыдущими опциями **-L**, а затем каталоги стандартного пути поиска библиотек (**-YP**). Если перед опцией **-l** установлено **-Bdynamic**, в каждом каталоге ищется **libстрока.so**, а затем **libстрока.a**; в противном случае ищется только библиотека **libстрока.a**.

-O Включает стадию оптимизации. Эта опция конфликтует с опцией **-g**, но имеет более высокий приоритет: если указаны обе опции, отладочная информация не генерируется. Эта опция также конфликтует с **-ql**, но имеет более низкий приоритет: стадия оптимизации пропускается, если указаны обе эти опции.

-o файл Заставляет компоновщик помещать результат в указанный файл, а не в **a.out**.

-p Вызывает генерацию дополнительного кода, считающего количество вызовов каждой подпрограммы. Если включена стадия компоновки, стандартный путь поиска библиотек (**-YP**) изменяется так, чтобы каталоги, которые должны содержать подготовленные так же библиотеки, проверялись до обычных каталогов. Более того, используются другие стартовые файлы, обеспечивающие запись (в файл **mon.out**) времени работы каждой подпрограммы; см. **prof(1)**.

-q строка Вызывает генерацию дополнительного кода, снабжающего программу определенными инструментальными средствами. Если **строка** - это **p**, результат аналогичен использованию опции **-p**. Если **строка** - это **l**, включается стадия базового профилирования блоков, генерирующая дополнительный код, считающий, сколько раз выполнялась каждая строка исходного кода; см. **lprof(1)**. Если **строка** - это **f**, создается *журнал профиля потока* (flow profile log); см. **fprof(1)**. Опция **-O** конфликтует с **-ql**, но имеет более низкий приоритет: стадия оптимизации не включается, если указаны обе эти опции.

-S Подавляет стадии ассемблирования и компоновки и выдает в результате файл на ассемблере (с суффиксом **.s**).

-v Вызывает выполнение на стадии компиляции дополнительных синтаксических, семантических и аналогичных выполняемым **lint(1)** проверок.

-Z строка Управляет упаковкой структур на стадии компиляции. Аргумент **строка** может иметь одно из следующих значений:

- **p1** Устанавливает выравнивание, по крайней мере, на границу байта для всех элементов структуры; или, другими словами, исключает заполнение. (Это можно также задать как **-Zp**.)
- **p2** Устанавливает выравнивание, по крайней мере, на границу двух байтов для элементов структуры, имеющих размер не менее двух байтов.
- **p4** Устанавливает выравнивание на два байта для двухбайтовых элементов структуры и выравнивание на границу четырех байтов для элементов большего размера. Эта установка используется по умолчанию.

Команда **cc** распознает **-e**, **-h**, **-u** и **-z** как опции стадии компоновки с параметрами. Эти и все остальные нераспознанные аргументы-опции передаются команде **ld(1)**.

Наконец, команда **cc** также распознает опцию **-#**. Если указана одна опция **-#**, команда **cc** будет выдавать каждое инструментальное средство с опциями и аргументами непосредственно перед вызовом. При наличии двух опций **-#** выдается также полный путь к каждому инструментальному средству; при указании трех таких опций все вызовы пропускаются.

ФАЙЛЫ

a.out
: стандартное имя создаваемого выполняемого файла

INCDIR
: последний каталог для поиска включаемых файлов

LIBDIR/*crt*.o
: объектные файлы стартового кода

LIBDIR/acomp
: препроцессор и компилятор

LIBDIR/optim
: оптимизатор

LIBDIR/basicblk
: базовый профилировщик блоков

BINDIR/as
: ассемблер

BINDIR/ld

компоновщик

LIBDIR/libc.so
: динамическая разделяемая версия стандартной библиотеки C

LIBDIR/libc.a
: архивная версия стандартной библиотеки C

DIR/libp
: подкаталог каждого элемента **LIBPATH**, в котором нужно искать библиотеки, поддерживающие профилирование

INCDIR
: обычно **/usr/include**

LIBDIR
: обычно **/usr/ccs/lib**

BINDIR
: обычно **/usr/ccs/bin**

LIBPATH
: обычно **/usr/ccs/lib:/usr/lib**

TMPDIR
: обычно **/var/tmp**, но может быть изменено путем задания переменной среды **TMPDIR**.

/usr/lib/locale/локаль/LC_MESSAGES/uxcds
: файл сообщений для текущего языка (См. **LANG** в **environ(5)**.)

ССЫЛКИ

as(1), **debug(1)**, **ld(1)**, **lint(1)**, **lprof(1)**, **monitor(3C)**, **prof(1)**, **tmpnam(3S)**

Kernighan, B. W., and Ritchie, D. M., The C Programming Language, Second Edition, Prentice-Hall, 1988

American National Standard for Information Systems - Programming Language C, X3.159-1989

International Standard ISO/IEC 9899:1990, Programming Languages - C

International Standard ISO/IEC 9945-1:1990, Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]

ПРИМЕЧАНИЯ

Если опция выбора режима (такая как **-Q** или **-X**) указана несколько раз, обычно используется последняя.

Для нового кода необходимо использовать **-Xa**, **-Khost** и **-v**; старый код лучше обрабатывается при указании **-Xt** и **-Kno_host**.

Для типичной многопользовательской системы на основе процессора Intel486, самый быстрый код, скорее всего, будет сгенерирован при указании опций **-Xa**, **-Kno_ieee**, **-Kinline** и **-O** и при использовании стандартных заголовочных файлов типа .