Руководство пользователя по редактору связей.

Данный документ описывает процесс и особенности компоновки (сборки) образа памяти исполняемой программы для мультиклеточного процессора.

Содержание

1 Общие сведения о редакторе связей **5**

2 Использование редактора связей **6**

3 Скрипты **7**

3.1 Выражения . . . 7
3.1.1 Константы . . . 7
3.1.2 Имена символов . . . 8
3.1.3 Счётчик текущего местоположения . . . 8
3.1.4 Операторы . . . 9
3.1.5 Оценка и вычисление выражения . . . 10
3.1.6 Секция результата вычисления выражения . . . 11
3.1.7 Встроенные функции . . . 12
3.1.7.1 ABSOLUTE . . . 13
3.1.7.2 ADDR . . . 13
3.1.7.3 ALIGN . . . 13
3.1.7.4 ALIGNOF . . . 14
3.1.7.5 ASSERT . . . 14
3.1.7.6 DEFINED . . . 15
3.1.7.7 LENGTH . . . 15
3.1.7.8 LOADADDR . . . 15
3.1.7.9 MAX . . . 15
3.1.7.10 MIN . . . 15
3.1.7.11 ORIGIN . . . 15
3.1.7.12 SEGMENT_START . . . 16
3.1.7.13 SIZEOF . . . 16
3.2 Операции присваивания . . . 16
3.2.1 Простые присваивания . . . 16
3.2.2 HIDDEN символы . . . 18
3.2.3 PROVIDE символы . . . 18
3.2.4 PROVIDE_HIDDEN символы . . . 18
3.3 Команды . . . 18
3.3.1 ASSERT . . . 18
3.3.2 DEBUG . . . 19
3.3.3 ENTRY . . . 20
3.3.4 GROUP . . . 20
3.3.5 INCLUDE . . . 20
3.3.6 INPUT . . . 20

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.7 MEMORY . . . 21
3.3.8 OUTPUT . . . 23
3.3.9 PHDRS . . . 23
3.3.10 REGION_ALIAS . . . 25
3.3.11 SEARCH_DIR . . . 25
3.3.12 SECTIONS . . . 25
3.3.12.1 Аттрибуты выходной секции . . . 26
3.3.12.1.1 Имя выходной секции. . . . 27
3.3.12.1.2 Виртуальный адрес (VMA) выходной секции. . . . 27
3.3.12.1.3 Тип выходной секции. . . . 27
3.3.12.1.4 Адрес загрузки (LMA) выходной секции. . . . 27
3.3.12.1.5 Принудительное выравнивание выходной секции. . . . 28
3.3.12.1.6 Принудительное выравнивание входной секции. . . . 28
3.3.12.1.7 Регион памяти выходной секции. . . . 29
3.3.12.1.8 Программный заголовок (сегмент) выходной секции. . . . 29
3.3.12.1.9 Заполнитель выходной секции. . . . 29
3.3.12.2 Описание входных секций . . . 30
3.3.12.2.1 Базовое описание входной секции. . . . 30
3.3.12.2.2 Шаблоны подстановки для входных секций . . . 31
3.3.12.2.3 Входная секция для общих символов. . . . 32
3.3.12.2.4 Пример описания входной секции. . . . 33
3.3.12.3 Данные выходной секции, непосредственно включаемые с помощью специальных команд . . . 33
3.3.12.4 Отбрасывание выходной секции . . . 34
3.3.13 STARTUP . . . 35

Все права защищены © ОАО «Мультиклет» 2010 — 2018

1 Общие сведения о редакторе связей

Компоновщик или редактор связей для мультиклеточного процессора осуществляет сборку образа исполняемой программы путём объединения данных одного или нескольких объектных файлов и, в случае необходимости, объектных файлов статических библиотек, а также связывание символов объектных файлов, так чтобы все ссылки на эти символы имели верные адреса времени выполнения.

Объектные файлы представляют собой результат компиляции ассемблером исходного кода программы.

Статические библиотеки представляют собой обычные архивы объектных файлов, для создания которых используется команда ar.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

2 Использование редактора связей

Представленный редактор связей запускается из командной строки командой ***mc-ld-s1***, аргументами которой являются один или несколько объектных файлов. Поддерживаются также следующие опции:

-l, --library=namespec — добавить файл архива namespec к списку файлов, используемых для сборки. Данная опция может использоваться многократно. Редактор связей будет искать путь к библиотеке для имени файла libnamespec.a. Поиск архивного файла осуществляется редактором связей только один раз, в том месте, где он указан в командной строке. Если в архиве определяется символ, который был неопределённым в каком-либо объектном файле и этот объектный файл указан в командной строке до файла архива, компоновщик добавит в результирующий файл образа соответствующие объектные файлы архива. В тоже время, наличие неопределённого символа в каком-либо объектном файле, указанного в командной строке после файла архива, не приведёт к повторному поиску символа в ранее указанном архивном файле. Любой архивный файл може быть указан в камандной строке несколько раз.

-L, --library-path=searchdir — добавить путь searchdir в список путей, в которых редактор связей будет искать статические библиотеки (архивные файлы, задаваемые опцией -l). Данная опция может использоваться многократно. Директории используются в том порядке, в котором они указаны в коммандной строке. Все значения опции -L применяются ко всем значениям опции -l, независимо от того, в каком порядке эти опции указаны.

-o, --output=FILE — поместить вывод в файл образа FILE. Если данная опция не используется, то имя выходного файла по умолчанию — image.bin.

-T, --script=FILE — прочитать файл скрипта управления редактором связей.

-M, --print-map — вывести в стандартный поток вывода информацию о размещении данных объектных файлов в памти и значениях, назначенных символам.

-V — напечатать номер версии редактора связей.

-h, --help — показать это сообщение и выйти.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3 Скрипты

Скрипты редактора связей предназначены для управления процессом сборки образа исполняемой пргограммы из одного или нескольких объектных файлов.

Скрипт представляет из себя набор команд. Каждая команда является либо ключевым словом, за которым могут следовать аргументы, либо присваиванием значения символу. Команды разделяются символом точки с запятой (';'). Пробельные символы игнорируются.

Строки, состоящие из символов латинского алфавита ([a..zA..Z]), точки (.), подчёркивания (_-), десятичных цифр ([0..9]), и не начинающиеся с цифры, могут быть заданы без заключения их в двойные кавычки ("). Все остальные строки должны быть заключены в двойные кавычки.

В скрипте можно использовать следующие типы комментариев:

- однострочный комментарий

  данный тип комментария начинается с '//' и заканчивается концом строки

- многострочный комментарий

  данный тип комментария начинается с '/*' и заканчивается '*/', т. е. все, что находится между этими ограничителями игнорируется; вложенные комментарии не допускаются.

В основном скрипты редактора связей используются для описания расположения секций входных объектных файлов в секциях выходного файла, а также распределения выходных секций в памяти целевой машины.

Для передачи редактору связей пользовательского скрипта необходимо использовать опцию командной строки '-T'. Если скрипт не был предоставлен пользователем, используется скрипт по умолчанию.

3.1 Выражения

Синтаксис выражений, использующихся в скриптах редактора связей, идентичен синтаксису выражений языка C. В выражениях могут быть использованы только целочисленные константы, ссылки на глобальные символы, а также встроенные функции.

3.1.1 Константы

Все константы целые.

Также, как и в C, редактор связей рассматривает целые числа, начинающиеся с '0', как восьмеричные, и целые, начинающиеся с '0x' или '0X', как шестнадцатиричные. Также допускается

Все права защищены © ОАО «Мультиклет» 2010 — 2018

использование суффиксов 'K' и 'M' для умножения констант на 1024 или 1024*1024 соответственно. Например, в следующем примере определяется одно и тоже значение:

fourk1 = 4K;

fourk2 = 4096;

fourk3 = 0x1000.

3.1.2 Имена символов

Без двойных кавычек имя символа может начинаться с буквенной литеры, подчёркивания, точки и содержать буквенно-цифровые символы, подчёркивание, точку. Имена символов не должны конфликтовать с ключевыми словами. Возможно определить символ, имя которого будет содержать недопустимые литеры, заключив его в двойные кавычки. Например, "with a space".

3.1.3 Счётчик текущего местоположения

Специальная переменная точка '.', которая может быть использована в выражениях, всегда содержит значение текущего местоположения выходной секции. Символ '.' можно использовать в выражении точно также, как и обычный символ.

Присваивание значения символу '.' вызовет перемещение счётчика текущего местоположения. Это может быть использовано для создания пропусков в выходной секции. Счётчик текущего местоположения не может быть уменьшен как внутри выходной секции, а также за её пределами, если это вызовет перекрытие згрузочных адресов (LMAs).

```
SECTIONS
{
 output : {
           file1(.text)
           . = . + 1000;
           file2(.text)
           . += 100;
           file3(.text)
 } = 0xFFAAFFAA;
}
```

В данном примере секция '.text' из файла file1 размещается в начале выходной секции 'output'. Далее следует пропуск размером 1000 байтов. Затем секция '.text' из файла file2, за которой также следует пропуск размером 100 байтов перед секцией '.text' из файла file3. Обозначение '= 0xFFAAFFAA' определяет записываемые в пропуски данные.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

Замечание: в действительности '.' ссылается на байтовое смещение внутри текущего объекта, в котором содержатся данные. Обычно это утверждение (statement) SECTIONS, чей начальный адрес 0, следовательно '.' может быть использована как абсолютный адрес. Если, тем не менее, '.' используется внутри описания секции, она ссылается на байтовое смещение от начала этой секции, а не на абсолютный адрес. Таким образом, в скрипте подобному ниже:

```
SECTIONS
{
 . = 0x100;
 .text: {
         *(.text)
         . = 0x200;
        }
 . = 0x1000;
 .data: {
         *(.data)
         . += 0x400;
        }
}
```

Стартовый адрес секции '.text' будет приравнен значнию 0x100, а размер будет в точности соответствовать 0x200 байтам, даже, если окажется недостаточно данных во входных секциях '.text' для заполнения указанного пространства. Если данных окажется слишком много, то это вызовет ошибку, так как будет осуществлена попытка уменьшить текущее значение счётчика текущего местоположения.

Стартовый адрес секции '.data' будет приравнен значнию 0x1000, а также будет иметь дополнительных 0x400 байтов пространтсва после всех данных из входных секций '.data' и перед концом самой выходной секции '.data'.

3.1.4 Операторы

Редактор связей распознаёт стандартный набор арифметических операторов языка C, со стандартными уровнями приоритета и связываниями:

Поддерживаемые операции в порядке снижения приоритета

Лексемы	**Операции**	**Приоритет**
symbol	символ	13
$func(...)$	вызов функции	12
$unary +$	унарный плюс	11
$unary -$	унарный минус	11

Все права защищены © ОАО «Мультиклет» 2010 — 2018

unary !	логическое НЕ	11
unary ~	побитовое НЕ	11
*, /, %	мультипликативные операции	10
+, −	аддитивные операции	9
<<, >>	сдвиг влево и вправо	8
<, >, <=, >=	отношения	7
==, ! =	равенство/неравенство	6
&	побитовое И	5
^	побитовое исключающее ИЛИ	4
|	побитовое ИЛИ	3
&&	логическое И	2
||	логическое ИЛИ	1
? :	условная операция	0

3.1.5 Оценка и вычисление выражения

Редактор связей оценивает выражения ленивым способом: он вычисляет значение выражения только тогда, когда это необходимо.

Редактору связей нужна некоторая информация: значение стартового адреса первой секции, начало и размер регионов памяти для того, чтобы в принципе осуществлять связывание. Эти значения вычисляются как можно раньше.

Однако, другие значения (напрмер, значения символов) не известны или не нужны до момента распределения памяти. Такие значения оцениваются позже, когда другая информация (например, размеры выходных секиций) станет доступна при присваивании символам значения выражения.

Размеры секций не могут быть известны до момента распределения памяти, поэтому присваивания, зависящие от распределения памяти, не выполняются.

Некоторые выражения, которые зависят от значения счётчика текущего местоположения '.', должны вычисляться во время распределения секций в памяти.

Если результат выражения необходим, но значение не доступно, тогда генерируется ошибка. Например, следующий скрипт

```
SECTIONS
{
 .text 9 + non_constatnt_symbol : { *(.text) }
```


Все права защищены © ОАО «Мультиклет» 2010 — 2018

```
}
```

вызовет сообщение об ошибке 'non constant expression'.

3.1.6 Секция результата вычисления выражения

Адреса и символы могут быть абсолютными, либо определены в какой-либо секции. Символы, связанные с секцией, являются перемещаемыми. С другой стороны, абсолютный символ всегда будет иметь одно и тоже значение на протяжении всех последующих операций связывания.

Некоторые члены выражения являются адресами. Это справедливо для символов, значение которых определено в какой-либо секции, а также для встроенных функций, которые возвращают адрес, таких как ADDR, LOADADDR, ORIGIN и SEGMENT_START. Другие члены выражения являются просто числами или встроенными функциями, результат которых не является адресом, например, LENGTH. В выражениях, которые используются за пределами описания выходной секции, все числа трактуются как абсолютные адреса. В выражениях, которые используются внутри описания выходной секции, все символы с абсолютным значением трактуются как числа.

В следующем примере

```
SECTIONS
{
 . = 0x100;
 __executable_start = 0x100;
 .data :
 {
  . = 0x10;
  __data_start = 0x10;
  *(.data)
 }
}
```

в первых двух присваиваниях . и __executable_start устанавливаются равными абсолютному адресу 0x100, затем в следующих двух присваиваниях . и __data_start устанавливаются равными 0x10 относительно секции .data.

Для оценки членов выражения, содержащих числа, относительные и абсолютные адреса, редактор связей использует следующие правила:

- для унарных операций над относительным адресом и бинарных операций над двумя относительными адресами в одной и той же секции или между относительным адресом и числом, оператор применяется к смещению части адреса(ов);

Все права защищены © ОАО «Мультиклет» 2010 — 2018

- для унарных операций над абсолютным адресом, бинарных операций над одним или несколькими абсолютными адресами или над двумя относительными адресами из различных секций, перед применением оператора осуществляется конвертация неабсолютного члена к абсолютному адресу.

Секция результата каждого подвыражения определяется следующим образом:

- результатом опрерации только над числами будет число;
- результатом операций сравнения '&&' и '||' будет число;
- результатом любой другой бинарной арифметической или логической операции над двумя относительными адресами в одной и той же секции или над двумя абсолютными адресами (после конвертирования) будет чило;
- результатом других операций над относительными адресами или над одним относительным адресом и числом, будет относительный адрес в той же самой секции, что и секция операнда(ов);
- результатом других операций над абсолютными адресами (после конвертирования) будет абсолютный адрес.

Для принудительного приведения результата выражения, который является относительным адресом, к абсолютному адресу необходимо использовать встроенную функцию ABSOLUTE. Например, чтобы создать абсолютный символ, значение которого равно адресу конца выходной секции '.data', необходимо написать в скрипте:

```
SECTIONS
{
 .data : { *(.data) _edata = ABSOLUTE(.); }
}
```

Если встроенная функция 'ABSOLUTE' не была использована, то значение '_edata' было бы относительным адресом в секции '.data'.

Использование встроенной функции LOADADDR также преобразует относительный адрес в абсолютный.

3.1.7 Встроенные функции

В выражениях скрипта редактора связей могут быть использованы встроенные функции.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.1.7.1 ABSOLUTE(expr)

Возвращает абсолютное (неперемещаемое) значение выражения exp. В основном используется для присваивания абсолютного значения символу внутри описания секции, где значения символов, как правило, являются относительными.

3.1.7.2 ADDR(section)

Возвращает адрес (VMA) именованной секции. Местоположение данной секции должно быть ранее определено в скрипте. В следующем примере символам start_of_output_1, symbol_1 и symbol_2 присваиваются эквивалентные значения, за исключением того, что символ symbol_1 будет относительным в секции .output1, а два других абсолютными:

```
SECTIONS
{
 .output1 :
 {
  start_of_output_1 = ABSOLUTE(.);
  ...
 }
 .output :
 {
  symbol_1 = ADDR(.output1);
  symbol_2 = start_of_output_1;
 }
}
```

3.1.7.3 ALIGN(align) / ALIGN(expr, align)

Возвращает значение счётчика текущего местоположения (.) или значение результата произвольного выражения, выравненное на следующую границу align. Одно операндный ALIGN не изменяет текущее значение счётчика текущего местоположения, а только выполняет арифметическую операцию и возвращает результат. Двух операндный ALIGN выполняет арифметическую операцию над результатом произвольного выражения exp и возвращает результат. (ALIGN(align) является эквивалентом ALIGN(., align)).

Далее следует пример, в котором адрес выходной секции .data устанавливается равным текущему значению счётчика текущего местоположения выровненным по границе 0x2000 байтов после предыдущей секции, а также устанавливается значение переменной A внутри секции равным значению выравнивания по границе 0x8000 байтов после всех входных секций:

Все права защищены © ОАО «Мультиклет» 2010 — 2018

```
SECTIONS
{
 ...
 .data ALIGN(0x2000) :
 {
  *(.data)
  A = ALIGN(0x8000);
 }
 ...
}
```

Первое использование встроенной функции ALIGN в этом примере определяет местоположение секции, так как результат данной функции используется в качестве необязательного адресного аттрибута при описании выходной секции. Следующее использование встроенной функции ALIGN в этом примере определяет значение символа A.

3.1.7.4 ALIGNOF(section)

Возвращает значение выравнивания в байтах именованной секции, если секция была распределена в памяти. Если секция не была распределена в памяти на момент выполнения данной функции, тогда генерируется сообщение об ошибке. В следующем примере, значение выравнивания выходной секции .output сохраняется в самой секции в качестве первого значения.

```
SECTIONS
{
 ...
 .output
 {
  LONG(ALIGNOF(.output))
  ...
 }
 ...
}
```

3.1.7.5 ASSERT(expr, msg)

Возвращает результат вычисления выражения expr, а также выводит сообщение об ошибке с текстом msg, если результат нулевой.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.1.7.6 DEFINED(symbol)

Возвращает 1, если символ находится в глобальной таблице символов редактора связей и определён в скрипте до использования данной функции, иначе возвращает 0. Данную функцию можно использовать для установки значения символа по умолчанию.

Следующий пример показывает, как установить значение глобального символа 'start', равным начальному адресу секции '.text', если данный символ ещё не определён:

```
SECTIONS
{
 ...
 .text :
 {
  start = DEFINED(begin) ? begin : . ;
  ...
 }
 ...
}
```

3.1.7.7 LENGTH(memory)

Возращает длину (размер) именованного региона памяти memory.

3.1.7.8 LOADADDR(section)

Возвращает абсолютный загрузочный адрес (LMA) именованной секции.

3.1.7.9 MAX(exp1, exp2)

Возвращает максимум из exp1 и exp2.

3.1.7.10 MIN(exp1, exp2)

Возвращает минимум из exp1 и exp2.

3.1.7.11 ORIGIN(memory)

Возвращает стартовый адрес именованного региона памяти memory.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.1.7.12 SEGMENT_START(segment, default)

Возвращает базовый адрес именованного сегмента. Если значение для сегмента было указано явно, используя опцию командной строки '-T', будет возвращено именно это значение, иначе будет возвращено значение по умолчанию.

3.1.7.13 SIZEOF(section)

Возвращает размер в байтах именованной секции, если секция была размещена. Если при выполнении данной функции секция не была размещена, редактор связей сгенерирует ошибку.

В следующем примере значения символов symbol_1 и symbol_2 устанавливаются равными одинаковым значениям:

```
SECTIONS
{
 ...
 .output
 {
  .begin = . ;
  ...
  .end = . ;
 }
 symbol_1 = .end - .begin;
 symbol_2 = SIZEOF(.output);
 ...
}
```

3.2 Операции присваивания

В скриптах допускается операция присваивания символу значения. Данная операция устанавливает значение символа и размещает его в таблице символов с глобальной областью видимости.

3.2.1 Простые присваивания

Для того, чтобы присвоить значение символу, необходимо использовать один из ниже перечисленных операторов:

$$symbol \ = \ expression;$$

Все права защищены © ОАО «Мультиклет» 2010 — 2018

symbol += *expression*;

symbol −= *expression*;

symbol ∗= *expression*;

symbol /= *expression*;

symbol <<= *expression*;

symbol >>= *expression*;

symbol &= *expression*;

symbol |= *expression*;

Первый оператор устанавливает значение символа, равным результату выражения. Во всех остальных случаях символ должен быть уже определён и его значение будет изменено согласно операции.

Каждая операция присваивания должна заканчиваться точкой с запятой (';').

В скриптах редактора связей символьные присваивания могут быть текстуально расположены только в трёх местах:

- вне команд, допустимых в скрипте редактора связей;
- внутри команды SECTIONS;
- внутри описания выходной секции в команде STCTIONS.

Секция, в которой определён символ, соответствует секции результата выражения.

Пример использования операции присваивания в трёх разных вариантах:

```
A = 0;
SECTIONS
{
 .text :
 {
  *(.text)
  end_of_text = .;
 }
 B = ALIGN(4);
 .data : { *(.data) }
}
```

В этом примере символ 'A' будет определён значением ноль. Значением символа 'end_of_-text' будет адрес, следующий за последней входной секцией '.text'. Значением символа 'B' будет адрес, следующий за выходной секцией '.text', выравненный на 4 байта.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.2.2 HIDDEN символы

С помощью команды HIDDEN определяется символ, который будет скрыт и не будет экспортирован. Синтаксис:

$$HIDDEN(symbol = expression)$$

3.2.3 PROVIDE символы

В некоторых случаях, желательно, чтобы символ был определён в скрипте только в том случае, если на него имеются ссылки и в тоже время он не был определён ни во одном объектном файле. Синтакс:

$$PROVIDE(symbol = expression)$$

3.2.4 PROVIDE_HIDDEN символы

В некоторых случаях, желательно, чтобы символ был определён в скрипте только в том случае, если на него имеются ссылки и в тоже время он не был определён ни во одном объектном файле. Синтакс:

$$PROVIDE_HIDDEN(symbol = expression)$$

Если данный символ будет определён, то он будет скрыт и не будет экспортирован.

3.3 Команды

В скриптах редактора связей поддерживается следующий набор команд:

- команда для установки точки входа (ENTRY);
- команды для работы с файлами (INCLUDE, INPUT, GROUP, OUTPUT, SEARCH_DIR, STARTUP);
- команды для определения регионов памяти (MEMORY, REGION_ALIAS);
- команда для описания выходных секций (SECTIONS);
- команда для определения программных сегментов (PHDRS);
- прочие команды (ASSERT, DEBUG, OUTPUT_ARCH).

3.3.1 ASSERT(expr, msg)

Команда ASSERT гарантирует, что результат вычисления выражения expr не нулевой. Если нулевой, редактор связей выводит сообщение об ошибке с текстом msg и завершает работу.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.2 DEBUG(fmt_str, expr_list)

Команда DEBUG предназначена для вывода результатов вычисления выражений, определённых списком expr_list, согласно заданной форматной строке fmt_str.

Данная команда может использоваться для отладки скрипта и никак не влияет на процесс сборки образа исполняемой программы.

Форматная строка fmt_str представляет из себя обычную строку символов, которая может содержать, а может и не содержать директивы: обычные символы, за исключением символов '%' и '\', которые печатаются в неизменном виде, escape–последовательности, начинающиеся с символа '\', а также номер результата выражения, начинающийся с символа '%', который необходимо вывести на печать. Номеруция результатов выражений начинается с единицы; максимальное значение номера — 9. Между символом '%' и номером результата выражения может быть указан один из следующих флагов:

- d — напечатать результат выражения в виде целого знакового десятичного числа;
- u — напечатать результат выражения в виде целого беззнакового десятичного числа;
- o — напечатать результат выражения в восьмеричной системе счисления;
- x, X — напечатать результат выражения в шестнадцатиричной системе счисления.

Поддерживаемые escape–последовательности:

- \a — звуковой сигнал (alert);
- \b — забой (backspace);
- \f — новая страница (formfeed);
- \n — перевод строки (newline);
- \r — возврат каретки (carriage return);
- \t — горизонтальная табуляция (horizontal tab);
- \v — вертикальная табуляция (verticalal tab);
- \\ — соответствует литералу '\'.

Для вывода на печать символа '%' в форматной строке fmt_str необходимо написать два символа '%' подряд ('%%').

В случае невалидного номера результата выражения (указанный номер результата превышает количество выражений в списке expr_list) или ошибки в форматной строке в соответствующем месте на печать будет выведено три вопросительных знака '???'.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.3 ENTRY(symbol)

Первая инструкция, с которой начинается выполнение программы называется точкой входа. Для того, чтобы установить точку входа, необходимо использовать команду ENTRY. Аргументом команды является имя символа symbol.

Так как процессор Multiclet S1 не поддерживает старт программы с произвольного адреса, то данная команда поддерживается парсером, но игнорируется кодогенератором, о чём редактор связей выводит соответствующее предупреждение.

3.3.4 GROUP(file, file, ...)

Команда GROUP указывает редактору связей интерпретировать перечисленные файлы как имена архивов (библиотек) и включить их в процесс сборки программы, как будто эти файлы были переданы в командной строке с помощью опции командной строки '-l'. Другими словами, команда GROUP подобна команде INPUT, за исключением того, что все файлы должны быть архивами (библиотеками).

Архивы (библиотеки) просматриваются неоднократно до тех пор, пока не останется ни одного неопределённого символа.

3.3.5 INCLUDE filename

Команда INCLUDE предназначена для влючения скрипта filename редактора связей в месте расположения данной команды. Поиск файла осуществляется в текущей директории, а также в любой из директорий, заданных опцией командной строки '-L'. Данную директиву можно использоавать на самом верхнем уровне, в командах MEMORY или SECTIONS, а также в описаниях выходной секции.

3.3.6 INPUT(file, file, ...)

Команда INPUT указывает редактору связей включить перечисленные файлы в процесс сборки программы, как будто эти файлы были переданы в командной строке.

Поиск файлов осуществляется в текущей директории; если файл не найден, то поиск файла будет осуществлён в директориях, заданых опцией командной строки '-L'.

Если использована команда 'INPUT (-lfile)', то имя будет преобразовано редактором связей в libfile.a, точно также, как при использовании опции командной строки '-l'.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.7 Команда MEMORY

Конфигурация редактора связей по умолчанию позваляет распределять всю доступную память. Используя команду MEMORY, данное поведение можно изменить.

Команда MEMORY описывает местоположение и размер блоков памяти целевой машины. Её можно использовать, чтобы описать какие регионы памяти могут быть использованы редактором связей, а какие стоит избегать. После определения регионов памяти их можно назначить выходным секиям. Редактор связей установит адрес секции основываясь на регионах памяти, а также выдаст предупреждения о переполнении региона памяти. Редактор связей не будет пытаться переставлять местами секции, чтобы уместить их в доступных регионах.

Скрипт редактора связей может содержать несколько команд MEMORY, однако в конечном итоге они будут объединены в одну. Допускается определять неограниченнное число регионов памяти внутри данной команды.

Синтаксис команды MEMORY:

```
MEMORY
{
 name [(attr)] : ORIGIN = origin, LENGTH = len
 ...
}
```

name — это имя, которое используется в скрипте для ссылки на данный регион памяти. Имя name региона не имеет никакого значения за пределами скрипта. Имена регионов сохраняются в отдельном пространстве имён, и никогда не будут конфликтовать с именами символов, файлов или секций. Каждый регион памяти должен иметь индивидуальное имя внутри команды MEMORY. Однако позже можно добавить алиас на имя существующего региона памяти, используя команду REGION_ALIAS.

Строка attr задаёт необязательный список аттрибутов, которые определяют использовать ли специфичные регионы памяти для входных секциий, размещение которых явно не указано в скрипте. Если для некоторой входной секции неопределена выходная секция, редактор связей создаст выходную одноимённую секцию. Если определены аттрибуты региона, редактор связей будет использовать их для выбора региона памяти, в котором будет создана выходная секция.

Строка attr должна состоять только из следующих символов:

'R' — read-only секция;

'W' — read/write секция;

'X' — исполняемая секция;

'A' — размещаемая секция;

Все права защищены © ОАО «Мультиклет» 2010 — 2018

'I' — инициализированная секция;

'!' — инверсия следующего аттрибута.

Если входная секция, размещение которой не определено в скрипте, соответствует любому из перечисленных аттрибутов, кроме '!', то она будет размещена в этом регионе памяти. Аттрибут '!' изменяет проверку на противоположную, поэтому секция будет размещена в данном регионе только, если она не соответствует ни одному из перечисленных аттрибутов.

origin — это числовое выражение, результат вычисления которого является начальным адресом региона памяти. Результат вычисления выражения должен быть константным, а само выражение не должно содержать никаких символов. Ключевое слово ORIGIN может быть сокращено до 'org' или 'o' (но не, например, ORG).

len — это числовое выражение, результат вычисления которого определяет размер в байтах региона памяти. Результат вычисления выражения должен быть константным, а само выражение не должно содержать никаких символов. Ключевое слово LENGTH может быть сокращено до 'len' или 'l'.

В следующем примере определяется два региона памяти, доступных для размещения: первый начинается с адреса 0 и имеет размер 128 килобайт, а второй начинается с адреса 0x400000 и имеет размер 256 килобайт. Редактор связей поместит в регион памяти rom каждую секцию, размещение которой явно не указано в скрипте и которая либо доступна только для чтения, либо исполняемая. Остальные секции редактор связей поместит в регион памяти ram.

```
MEMORY
{
 rom (rx)  : ORIGIN = 0, LENGTH = 128K
 ram (!rx) : org = 0x400000, l = 256K
}
```

Как только регион памяти определён, можно указать редактору связей разместить специфичные выходные секции в этом регионе, используя '>region'. Например, если определён регион памяти с именем mem_reg, необходимо использовать '>mem_reg' в описании выходной секции. Если для выходной секции неопределён адрес, редактор связей установит его значение равным первому доступному адресу в регионе памяти. Если размер всех включённых в регион памяти выходных секций превышает размер данного региона памяти, редактор связей выдаст сообщение об ошибке.

Для того, чтобы получить начальный адрес и размер региона памяти, в выражении необходимо использовать функции $ORIGIN(memory)$ и $LENGTH(memory)$ соответвственно, например, $addr_of_ram = ORIGIN(ram)$;.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.8 OUTPUT(filename)

Команда OUTPUT определяет имя выходного файла. Использование OUTPUT(filename) в скрипте в точности совпадает с использованием опции командной строки '-o filename'. Если используются оба варианта, предпочтение отдаётся опции командной строки.

3.3.9 Команда PHDRS

Формат ELF объектного файла использует программные заголовки, котороые описывают так называемые сегменты, а именно как следуетт размещать программу (сегменты) в памяти.

При загрузке ELF программы загрузчик читает программные заголовки для того, чтобы выяснить, как загружать программу. Это работает только в том случае, когда программные заголовки выставлены корректно.

Редактор связей создаёт обоснованные программные заголовки по умолчанию. Однако, в некоторых случаях, может возникнуть необходимость определить программные загодоловки более точно. В этом случае необходимо использовать команду PHDRS. При обнаружении команды PHDRS в скрипте, редактор связей не будет создавать никакие другие программные заголовки, крме явно определённых.

Синтаксис команды PHDRS.

```
PHDRS
{
 name type [ FILEHDR ] [ PHDRS ] [ AT ( address ) ] [ FLAGS ( flags ) ] ;
}
```

PHDRS, FILEHDR, AT, FLAGS являются ключевыми словами.

name используется только для ссылок внутри команды SECTIONS в скрипте и никак не проявляется в выходном файле. Имена программных заголовков сохраняются в отдельном пространстве имён и никак не конфликтуют с именами символов, файлов и секций. Каждый программный заголовок должен иметь уникальное имя. Заголовки обрабаьываются в порядке возрастания загрузочного адреса сопостовляемых секций.

Некоторые типы программных заголовков описывают сегменты памяти, которые загрузчик будет загружать из файла. В скрипте определяется содержимое данных сегментов путём размещения распределяемых (allocatable) выходных секций в сегментах. Для размещения секции в специфическом сегменте можно использовать аттрибут ':phdr' выходной секции.

Допустимо размещать некоторые секции в нескольких сегментах. Это просто означает, что один сегмент памяти содержит другой. Можно однократно использовать ':phdr' для каждого сегмента, который должен содержать секцию.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

Если разместить секцию более, чем в одном сегменте, используя ':phdr', то редактор связей разместит все последующие распределяемые секции, для которых не определён ':phdr', в том же самом сегменте. Это сделано для удобства, так как обычно весь набор соприкасающихся секций будет размещаться в одном сегменте.

Ключевые слова FILEHDR и PHDRS после программного заголовка поддерживаются парсером, но игнорируются кодогенератором.

Тип сегмента может принимать следующие значения (номер указывает значение ключевого слова).

PT_NULL (0) — неиспользуемый программный заголовок;

PT_LOAD (1) — данный сегмент должен быть загружен из файла;

PT_DYNAMIC (2) — в данном сегменте следует искать информацию о динамическом связывании (не поддерживается);

PT_INTERP (3) — в данном сегменте следует искать информацию об имени интерпретатора программы;

PT_NOTE (4) — в данном сегменте содержатся примечания;

PT_SHLIB (5) — зарезервированный тип программного заголовка;

PT_PHDR (6) — в данном сегменте содержится таблица программных заголовков;

expression — результатом вычисления выражения является числовое значение программного заголовка. Может быть использовано для типов не описанных выше.

Используя 'AT expression', можно определить конкретный адрес, по которому сегмент должен быть загружен в память. Аналогично аттрибуту AT, который используется в описании выходной секции. Команда AT программного заголовка переписывает значение аттрибута выходной секции.

Обычно редактор связей устанавливает флаги сегментов, основываясь на флагах секций, которые являются содержимым данного сегмента. Для явного определения флагов сегмента, можно использовать ключевое слово FLAGS. Значение флага должно быть целым числом. Оно используется в качестве значения поля p_flags программного заголовка.

Пример.

```
PHDRS
{
 text PT_LOAD;
 data PT_LOAD;
}
```


Все права защищены © ОАО «Мультиклет» 2010 — 2018

```
SECTIONS
{
 .text : { *(.text) } :text
 .data : { *(.data) } :data
}
```

3.3.10 REGION_ALIAS(alias, region)

Команда REGION_ALIAS предназначена для добавления алиасов на существующие регионы памяти, которые ранее были созданы командой MEMORY. Каждый алиас может соответствовать только одному региону памяти.

Команда REGION_ALIAS создаёт алиас alias для региона памяти region, что позваляет гибко распределять выходные секции в регионах памяти.

3.3.11 SEARCH_DIR(path)

Команда SEARCH_DIR добавляет путь к списку путей, где редактор связей осуществляет поиск библиотек. Использование SEARCH_DIR(path) в точночти совпадает с использованием опции командной строки '-L path'. Если используются оба варианта, тогда и редактор связей использует оба пути. Пути, заданные с использованием опции командной строки, используются первыми.

3.3.12 Команда SECTIONS

Команда SECTIONS указывает, как редактору связей осуществлять расположение входных секций в выходных, и как разместить выходные секции в памяти.

Формат команды SECTIONS:

```
SECTIONS
{
 sections-command
 sections-command
 ...
}
```

Каждая sections-command может быть:

- командами ASSERT, DEBUG, ENTRY;
- символьным присваиванием;

Все права защищены © ОАО «Мультиклет» 2010 — 2018

- описанием выходной секции.

Команды ASSERT, DEBUG, ENTRY и символьное присваивание допускается использовать внутри рассматриваемой команды для удобства использования счётчика местоположения. Это облегчает понимание скрипта, потому что данные команды могут быть использованы в значимых точках при описании макета выходного файла.

Если в скрипте не используется команда SECTIONS, редактор связей разместит каждую входную секцию в одноимённой выходной секции в порядке их размещения во входных файлах. Если, например, все входные секции представлены в первом входном файле, порядок секций в выходном файле будет соответствовать порядку их следования в первом входном файле. Первая секция будет размещена с нулевого адреса.

Синтаксис описания выходной секции:

```
section_name [address] [(type)] :
  [AT(lma)]
  [ALIGN(section_align)]
  [SUBALIGN(subsection_align)]
  {
   output-section-command
   output-section-command
   ...
  } [>region] [AT>lma_region] [:phdr :phdr ...] [=fillexp]
```

Двоеточие и круглые скобки в описании выходной секции являются обязательными. Символы перевода строки и остальные пробелы опциональны.

Каждая из output-section-command может быть:

- символьным присваиванием;
- описанием входной секции;
- данными, которые включаются непосредственно с помощью специальных команд.

3.3.12.1 Аттрибуты выходной секции

Ниже описаны аттрибуты выходной секции, которые могут быть определены при описаннии выходной секции.

Как правило, большинство из опциональных аттрибутов (указаны в квадратных скобках []) не используется.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.12.1.1 Имя выходной секции. section_name — имя выходной секции. Если имя выходной секции section_name содержит недопустимые символы, его необходимо заключить в двойные кавычки.

3.3.12.1.2 Виртуальный адрес (VMA) выходной секции. **address** — выражение, результат вычисления которого является адресом (VMA) выходной секции.

Выражение, определяющее адрес, опционально, а в случае его явного указания адрес выходной секции будет в точности ему соответствовать (без учёта значений выравнивания входных секций, содержащихся в выходной секции).

Если выходной адрес не определён, тогда адрес будет вычислен согласно следующему алгоритму:

- если для секции определён выходной регион памяти, то секция будет добавлена в этот регион и её адрес будет равен первому свободному адресу данного региона памяти;
- если была использована команда MEMORY для создания списка регионов памяти, тогда будет выбран первый регион из списка, атрибуты которого совместимы с аттрибутами выбранной секции; выходной адрес секции будет равен следующему свободному адресу в регионе;
- если регионы памяти не были определены, либо ни один из определённых регионов памяти не является подходящим для размещения секции, то выходной адрес секции будет вычислен на основе текущего значения счётчика местоположения.

Вычисленный адрес также будет выровнен согласно аттрибуту выравнивания выбранной выходной секции. Значение выравнивания выходной секции будет самым строгим (максимальным) среди всех значений выравнивания входных секций, содержащихся в выходной секции.

Определение адреса выходной секции вызывает изменение значения счётчика местоположения для непустых секций (пустые секции игнорируются).

3.3.12.1.3 Тип выходной секции. Каждая выходная секция может иметь тип. Тип — это ключевое слово в круглых скобках {(type)}. Допустимо использование следующих типов: NOLOAD (секция должна быть помечена как незагружаемая, т. е. она не будет загружаться в память при запуске программы).

Данный аттрибут поддерживается парсером редактора связей, однако его значение игнорируется.

3.3.12.1.4 Адрес загрузки (LMA) выходной секции. Каждая секция имеет виртуальный адрес (VMA) и загрузочный адрес (LMA). Виртуальный адрес описан выше. Загрузоч-

Все права защищены © ОАО «Мультиклет» 2010 — 2018

ный адрес определяется ключевым словом 'AT' или 'AT>'. Определение загрузочного адреса опционально.

Ключевре слово 'AT' в качестве аргумента требует выражение, результат вычисления которого точно определяет загрузочный адрес секции. Ключевое слово 'AT>' в качестве аргумента требует имя региона памяти, при этом загрузочный адрес секции устанавливается равным следующему свободному адресу региона, выровненному, в случае необходимости, согласно значению выравнивания секции.

Если ни одно из ключевых слов 'AT' или 'AT>' не задано для размещаемой секции, редактор связей будет использовать следующую процедуру определения загрузочного адреса секции:

- если задан конкретный виртуальный адрес секции (VMA), тогда он используется в качестве загрузочного адреса (LMA);
- если для секции определён выходной регион памяти, тогда загрузочный адрес (LMA) равен виртуальному адресу (VMA);
- если секция не является размещаемой, тогда загрузочный адрес (LMA) равен виртуальному адресу (VMA);
- если найден регион памяти, подходящий для размещения текущей секции, и этот регион уже содержит хотя бы одну секцию, тогда загрузочный адрес (LMA) утанавливается таким образом, чтобы разность между VMA и LMA совпадала с разностью между VMA и LMA последней размещённой секции в данном регионе;
- если не было объявлено ни одного региона памяти, тогда используется регион памяти, который охватывает всё адресное пространство, а загрузочный адрес (LMA) устанавливается согласно алгоритму, описанному на предыдущем шаге;
- если подходящий для размещения секции регион памяти не был найден или не было предыдущей секции в найденном регионе, тогда LMA устанавливается равным VMA.

3.3.12.1.5 Принудительное выравнивание выходной секции. Для определения минимального значения выравнивания выходной секции необходимо использовать аттрибут ALIGN, аргументом которого является выражение. Результат вычисления выражения должен быть неотрицательным целым числом кратным степени 2, за исключением значения '–1', которое отменяет выравнивание.

3.3.12.1.6 Принудительное выравнивание входной секции. Для определения минимального значения выравнивания входной секции внутри выходной секции необходимо использовать аттрибут SUBALIGN, аргументом которого является выражение. Результат вычисления выражения должен быть неотрицательным целым числом кратным степени 2, за исключением значения '–1', которое отменяет выравнивание.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.12.1.7 Регион памяти выходной секции. Для того, чтобы назначить секции ранее определённый регион памяти необходимо использовать '>region'. Например:

```
MEMORY
{
 rom : ORIGIN = 0x1000, LENGTH = 0x1000
}

SECTIONS
{
 ROM : { *(.text) } >rom
}
```

3.3.12.1.8 Программный заголовок (сегмент) выходной секции. Для того, чтобы назначить секцию ранее определённому программному сегменту необходимо использовать ':phdr'. Если секция назначена одному или более сегменту, то все последующие размещённые секции будут назначены также тем сегментам, пока не будет явного указания модификатора ':phdr'. Пример:

```
PHDRS
{
 text PT_LOAD;
}

SECTIONS
{
 .text : { *(.text) } :text
}
```

3.3.12.1.9 Заполнитель выходной секции. Для того, чтобы установить шаблон заполнителя для всей секции, необходимо использовать '=fillexp'. fillexp является выражением. Любые неопределённые регионы памяти внутри выходной секции будут заполнены этим значением.

Последовательность байт в числе, которым является результат вычисления выражения fillexp, — big-endian.

Шаблон заполнителя можно изменить, используя команду FILL внутри команд выходной секции. Пример:

```
SECTIONS
{
 .text : { *(.text) } =0x90909090
```


Все права защищены © ОАО «Мультиклет» 2010 — 2018

```
}
```

3.3.12.2 Описание входных секций

Наиболее часто используемая команда выходной секции — это команда описания входной секции.

Описание входной секции является самой базовой командой скрипта редактора связей. Описания выходных секций используются для указания редактору связей карты расположения программы в памяти. Описания входных секций используются для указания редактору связей карты расположения входных секций в выходных секциях.

3.3.12.2.1 Базовое описание входной секции. Описание входной секции состоит из имени файла, за которым опционально следует список имён входных секций в круглых скобках.

Имя файла и имя секции может быть задано в виде шаблона имён, которые будут описаны позже.

Наиболее общее описание входной секции указыает редактору связей на включение всех входных секций с определённым именем в (одноимённую) выходную секцию. Например, для того, чтобы включить все входные секции '.text', необходимо написать:

```
*(.text)
```

Здесь символ '*' является шаблоном, который соответствует любому имени файла. Для того, чтобы исключить список файлов из соответсвия имени файла шаблону можно использовать команду EXCLUDE_FILE. Например:

```
*(EXCLUDE_FILE (file1.o *_xx.o) .data)
```

означает включить все секции .data из всех файлов, за исключением секций .data из файлов file1.o и *_xx.o (* — шаблон, соответствующий любой последовательности символов).

Существует два способа включить более, чем одну входную секцию в выходную:

```
*(.text .rdata)
*(.text) *(.rdata)
```

Разница между двумя представленными вариантами в порядке включения входных секций '.text' и '.rdata' в выходную секцию. В первом случае, они будут смешаны, т. е. расположены в выходной секции в порядке их появления на входе редактора связей. Во втором случае, сначала будут расположены все входные секции '.text', за которыми будут следовать все входные секции '.rdata'.

Для того, чтобы включить входные секции из определённых файлов, необходимо явно указать их имена. Например:

Все права защищены © ОАО «Мультиклет» 2010 — 2018

```
data.o(.data)
```

Для того, чтобы включать входные секции на основе их флагов, необходимо использовать команду INPUT_SECTION_FLAGS. Например:

```
SECTIONS
{
 .text1 : { INPUT_SECTION_FLAGS(SHF_MERGE & SHF_STRINGS) *(.text) }
 .text2 : { INPUT_SECTION_FLAGS(!SHF_WRITE) *(.text) }
}
```

В этом примере, выходная секция '.text1' будет содержать любую из одноимённых входных секций, флаги SHF_MERGE и SHF_STRINGS которых будут установлены. Выходная секция '.text2' будет содержать любую из одноимённых входных секций, флаг SHF_WRITE которых будет сброшен.

Для указания файлов внутри архива необходимо использовать шаблон имени архива, затем символ ':', а затем шаблон имени файла, БЕЗ пробельных символов вокруг символа ':' (при этом полученная строка должна быть заключена в двойные кавычки):

- "archive:file" — соответствует файлу file внутри архива libarchive.a;
- "archive:" — соответствует всем файлам внутри архива libarchive.a;
- ":file" — соответствует файлу file, который не найден ни в одном из архивов. Другими словами, файл file НЕ должен быть частью архива, т. е. это обычный объектный файл.
- "file" — соответствует файлу, который может быть как частью любого архива, так и обычным объектным файлом.

Как имя архива, так и имя файла, может содержать шаблон имени. "archive:file" может быть использован внутри списка команды EXCLUDE_FILE, но не может присутствовать в других контекстах скрипта редактора связей. Например, недопустимо извлекать файл из архива, используя "archive:file" внутри команды INPUT.

В случае использования имени файла БЕЗ списка секций, все секции входного файла будут включены в выходную секцию. Например:

```
data.o
```

3.3.12.2.2 Шаблоны подстановки для входных секций В описании входной секции, как в имени файла, так и в имени секции, можно использовать шаблоны подстановки. Так, например, имя файла '*' означает любое имя файла.

Шаблоны подстановки:

'*' — соответствует любому количеству любых символов (ноль и более);

Все права защищены © ОАО «Мультиклет» 2010 — 2018

'?' — соответствует любому одному символу;

'[chars]' — соответствует одному символу из набора символов, символ '-' используется для задания диапазона символов, например, '[a-z]' соответствует любому латинскому символу в нижнем регистре;

'\' — экранирует следующий символ.

Когда имя файла соответствует шаблону, символы подстановки не будут соответствовать символу '/' (используется для разделения имён директорий в Unix). Шаблон, состоящий из одного символа '*' является исключением; он всегда соответствует любому имени файла, вне зависимости от того, содержит имя файла символ '/' или нет. В имени секции символы подстановки будут соответствовать символу '/'.

Шаблоны имени файла применяются только к файлам, которые явно определены в командной строке или в команде INPUT. Редактор связей не осуществляет поиск директорий для расширения шаблонов.

Если имя файла соответствует более, чем одному шаблону, или имя файла задано явно и при этом также соответствует какому-либо шаблону, то редактор связей будет использовать первое совпадение в скрипте. Например, последовательность описаний входных секций, приведённая ниже, возможно является ошибочной, так как правило data.o никогда не будет использовано:

```
.data : { *(.data) }
.data1 : { data.o(.data) }
```

Следующий пример показывает, как шаблоны подстановки могут быть использованы для разделения файлов. Скрипт указывает редактору связей разместить все входные секции '.text' в выходной секции '.text' и все входные секции '.bss' в выходной секции '.bss'. Редактор связей разместит входные секции '.data' из входных файлов, имя которых начинается с заглавной буквы, в выходной секции '.DATA'; для всех других файлов, редактор связей разместит входные секции '.data' в выходной секции '.data'.

```
SECTIONS
{
 .text : { *(.text) }
 .DATA : { [A-Z]*(.data) }
 .data : { *(.data) }
 .bss : { *(.bss) }
}
```

3.3.12.2.3 Входная секция для общих символов. Общие символы во входных файлах будут размещены в секции '.bss' выходного файла.

Все права защищены © ОАО «Мультиклет» 2010 — 2018

3.3.12.2.4 Пример описания входной секции. Следующий скрипт указывает редактору связей прочитать все секции из входного файла all.o и разместить их в начале выходной секции 'outputA', которая начинается с адреса 0x10000. Все секции '.input1' из входного файла file1.o разместить непосредственно за всеми секциями входного файла all.o в той же самой выходной секции 'outputA'. Все секции '.input2' из входного файла file1.o размещаются в выходной секции 'outputB, за которыми сразу следуют все входные секции '.input1' из входного файла file2.o. Все остальные входные секции '.input1' и '.input2' из любых других входных файлов размещаются в выходной секции 'outputC'.

```
SECTIONS
{
 outputA 0x10000 :
 {
  all.o
  file1.o (.input1)
 }
 outputB :
 {
  file1.o (.input2)
  file2.o (.input1)
 }
 outputC :
 {
  *(.input1)
  *(.input2)
 }
}
```

3.3.12.3 Данные выходной секции, непосредственно включаемые с помощью специальных команд

В выходную секцию можно явно включить байты данных, используя команды выходной секции BYTE, SHORT, LONG, QUAD. За каждым ключевым словом следует выражение в круглых скобках, результат вычисления которого и будет сохранён в выходной секции. Значение сохраняется по текущему местоположению счётчика текущего местоположения.

Команды BYTE, SHORT, LONG и QUAD записывают в выходную секцию соответственно один, два, четыре и восемь байтов. После записи байтов, счётчик текущего местоположения увеличивается на соответствующее количество.

Например, ниже следующие команды приведут к последовательной записи в выходную секцию

Все права защищены © ОАО «Мультиклет» 2010 — 2018

одного байта, равным значению 0x01, и четырёх байтов, равных значению символа addr:

```
BYTE(1)
LONG(addr)
```

Порядок следования байт при выполнении комад BYTE, SHORT, LONG, QUAD определяется целевой архитектурой.

Описанные выше команды могут быть использованы только внутри описания выходной секции, поэтому следующий пример вызовет ошибку:

```
SECTIONS
{
 .text : { *(.text) }
 LONG(1)
 .data : { *(.data) }
}
```

в то время как следующий будет правильно отработан:

```
SECTIONS
{
 .text : { *(.text) LONG(1) }
 .data : { *(.data) }
}
```

Для определения шаблона заполнителя для текущей выходной секции необходимо использовать команду FILL, за которой следует выражение в круглых скобках. Любые неопределённые регионы памяти внутри данной выходной секции (например, пропуски, вызванные выравниванием входных секций) заполняются значением выражения, которое повторяется необходимое количество раз. Допускается использование различных шаблонов заполнителя в различных частях выходной секции.

Пример заполнения неопределённых промежутков значением 0x90:

```
FILL(0x90909090)
```

Команда FILL схожа с аттрибутом '=fillexp' описания выходной секции, за исключением того, что она применяется к части секции, которая следует за ней ('=fillexp' применяется ко всей секции). Если одновременно используютя два варианта, предпочтение отдаётся команде FILL.

3.3.12.4 Отбрасывание выходной секции

Редактор связей не будет создавать секции с пустым содержимым (без данных). Это сделано для возможности описания входных секций в скрипте, которые могут или НЕ могут присут-

Все права защищены © ОАО «Мультиклет» 2010 — 2018

ствовать во входных файлах. Например:

```
.other\_data : { *(.other\_data\_1 .other\_data\_2) }
```

выходная секция '.other_data' будет создана в выходном файле только в том случае, если имеется хотя бы одна непустая (с данными) входная секция '.other_data_1' или '.other_-data_2' в любом из входных файлов.

3.3.13 STARTUP(filename)

Команда STARTUP подобна команде INPUT, за исключением того, что заданный файл будет первым в списке обрабатываемых файлов вне зависимости от места положения данной команды.

Все права защищены © ОАО «Мультиклет» 2010 — 2018