armlink 第三章 scatter 語法（二）

原創 安仔都有人用 2020-04-14 16:37

第三章 scatter 語法（二）表達式與內置函數

scatter 文件通常包含有數字常量。這些常量可以直接指定，也可以通過對表達式求值而得到

3.1 表達式中可用的運算符

+，-，*，/,AND,OR,()

AND 和 OR 的運算規則和 C 語言的一樣。

例如：

```
#define BASE_ADDRESS 0x8000
#define ALIAS_NUMBER 0x2
#define ALIAS_SIZE 0x400
#define AN_ADDRESS (BASE_ADDRESS+(ALIAS_NUMBER*ALIAS_SIZE))
```

scatter 文件可能包含如下代碼：

```
LOAD_FLASH AN_ADDRESS
```

預處理之後，它等價於

```
LOAD_FLASH ( 0x8000 + (0x2 * 0x400))
```

計算之後，為：

```
LOAD_FLASH 0x8800
```

3.2 表達式使用場景

表達式可以使用在如下的地方：

- 加載和執行 region 的基址
- 加載和執行 region 的偏移
- 加載和執行 region 的 max_size
- ALIGN,FILL 或者 PADVALUE 的參數
- ScatterAssert 函數的參數

例如：

```
LR1 0x8000 (2 * 1024)
{
    ER1 +0 (1 * 1024)
    {
        *(+RO)
    }
    ER2 +0 (1 * 1024)
    {
        *(+RW +ZI)
    }
}
```

3.3 表達式規則

表達式遵循 C 預處理規則

表達式由下面組成：

- 十進制或者十六進制的數字
- 算術運算符：+ ，-， /， *， ~， OR， AND。OR 和 AND 對應於 c 語言中的|和&
- 邏輯運算符：LOR,LAND,！。 LOR 和 LAND 對應於 c 語言中的||和&&
- 關係運算符：< ,<=,>,>=,== 。關係運算表達式如果為 true，返回非零；否則返回零
- 條件表達式：Expression?Expression1:Expression2。
  這個對應於 c 語言的條件運算符
- 函數返回數字

所有運算符在含義和優先級上都與 C 對應的運算符匹配

表達式不區分大小寫。

3.4 用於計算執行地址的內置函數

```
ImageBase(region_name)   //返回 region_name 的基址
ImageLenth(region_name)  //返回 region_name 的長度
ImageLimit(region_name)  //返回 region_name 的最後的地址
```

region_name 可以是加載或者執行 region 的名字。

注意：當使用.ANY 時，不能使用這些函數。因為該 region 只有在.ANY 分配好之後，才能知道大小

例子：

```
LR1 0x8000
{
    ER1 0x100000
    {
        *(+RO)
    }
}
LR2 0x100000
    {
        ER2 (ImageLimit(ER1)) ; 在 ER1 結束之後放置 ER2
        {
            *(+RW +ZI)
        }
}
```

3.5 ScatterAssert 函數和加載地址相關的函數

```
ScatterAssert(expression)   //如果 expression 返回 false，鏈接器就報錯
LoadBase(region_name) //region_name 的基址
LoadLength(region_name)   //region_name 的長度
LoadLength(region_name)   //region_name 的結束地址
```

例子如下：

```
LR1 0x8000
{
    ER0 +0
```

```
    {
        *(+RO)
    }
    ER1 +0
    {
        file1.o(+RW)
    }
    ER2 +0
    {
        file2.o(+RW)
    }
    ScatterAssert((LoadLength(ER1) + LoadLength(ER2)) < 0x1000);
LoadLength is compressed size
    ScatterAssert((ImageLength(ER1) + ImageLength(ER2)) < 0x2000);
ImageLength is uncompressed size
}
ScatterAssert(ImageLength(LR1) < 0x3000); Check uncompressed size of
load region LR1
```

3.6 符號相關的函數

```
defined(global_symbol_name)  //如果 global_symbol_name 沒有定義，則返回零，否則返回非零
```

例如：

```
LR1 0x8000
{
    ER1 (defined(version1) ? 0x8000 : 0x10000)
    {
        *(+RO)
    }
    ER2 +0
    {
        *(+RW +ZI)
    }
}
```

3.7 AlignExpr(expr,align)函數

`AlignExpr(expr,align)` //增加 expr 直到與指定的 align 邊界對齊，函數的計算方式為：(expr + (align-1)) & ~(align-1))

例如：

```
ER +0
{
    …
}
ER2 AlignExpr(+0x8000,8)
{
    …
}
```

3.8 GetPageSize() 函數

`GetPageSize()` //返回頁大小，本系列旨在裸機開發，此函數不做過多介紹

3.9 SizeOfHeaders()函數

`SizeOfHeaders()` //返回 ELF header 和 Programe Header table 的大小

3.10 最後來兩個栗子

最後的最後，來兩個使用函數的例子：
第一個例子，使用 AlignExpr 和 ImageLimit 緊密排列執行 region：

注意：下面例子中使用的

```
#! armcc -E
```

表示使用的預處理名字為 armcc,對應的預處理參數為-E,這將會在下一章中引入

`InRoot$$Sections`

表示 root region 的所有 section，這也會在下一章中引入介紹

```
#! armcc -E
#define START_ADDRESS 0x100000
#define PAGE_ALIGNMENT 0x100000
```

```
LR1 0x8000
{
    ER0 +0
    {
        *(InRoot$$Sections)
    }
    ER1 START_ADDRESS
    {
        file1.o(*)
    }
    ER2 AlignExpr(ImageLimit(ER1), PAGE_ALIGNMENT)
    {
        file2.o(*)
    }
    ER3 AlignExpr(ImageLimit(ER2), PAGE_ALIGNMENT)
    {
        file3.o(*)
    }
}
```

第二個例子:

有些時候可能想在第一個加載 region 中放置 ZI，然後在二個加載 region 中使用相對偏移來指定地址。你可能如下寫法：

```
LR1 0x8000
{
    er_progbits +0
    {
        *(+RO,+RW) ; 在加載 region 中佔空間
    }
    er_zi +0
    {
        *(+ZI) ; 在加載 region 中不佔空間
    }
}
LR2 +0 ;LR2 緊跟 LR1
{
    er_moreprogbits +0
    {
        file1.o(+RO) ; 在加載 region 中佔空間
```

```
    }
}
```

但是，鏈接器不會根據 ZI 來調整 LR2 的基址，所以這會導致 er_zi 覆蓋 er_moreprogbits，鏈接會產生一個錯誤。

為了糾正這個錯誤，使用 ImageLimit 函數，如下：

```
LR1 0x8000
{
    er_progbits +0
    {
        *(+RO,+RW)
    }
    er_zi +0
    {
        *(+ZI)
    }
}
LR2 ImageLimit(er_zi)
{
    er_moreprogbits +0
    {
        file1.o(+RO)
    }
}
```

本章完，下一章為 scatter 的具體使用

包括如何在指定地址放置變量，函數，如何映射寄存器等。