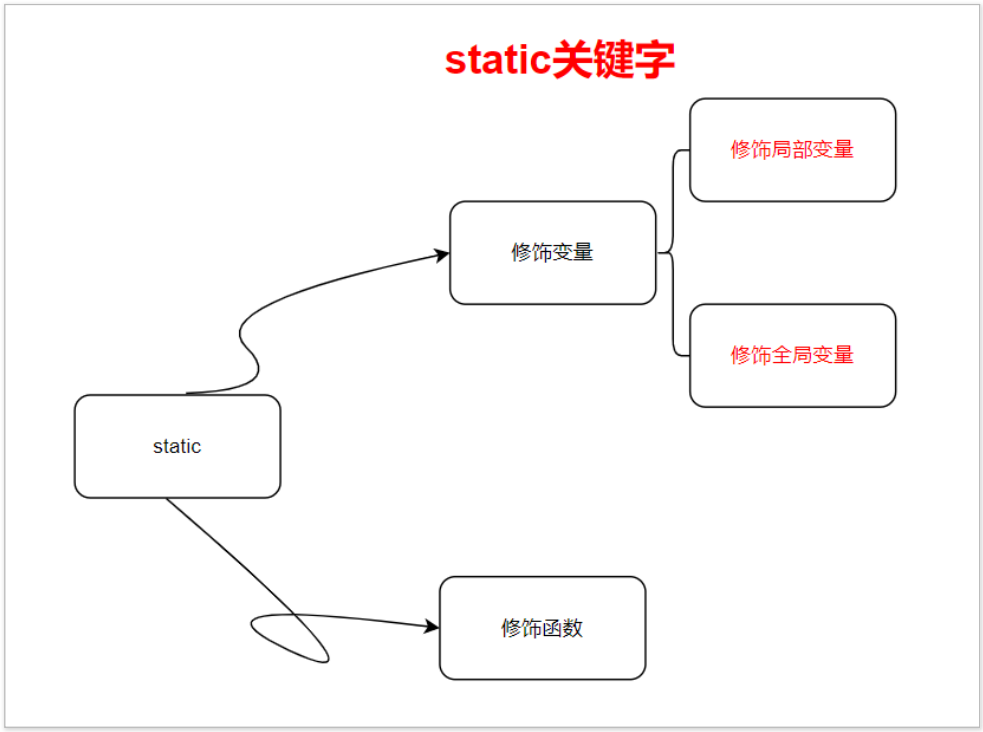
3.2 C 语言中的 static 的使用_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.2 C 语言中的 static 的使用涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

2.C 语言中的 static 的使用

static 在 C 语言中属于比较常见的关键字。它的用法很多，在一定的环境下使用，可以提高程序的运行性能。优化程序的结构。本文中主要是对 C 语言中静态关键字 static 进行讲解。



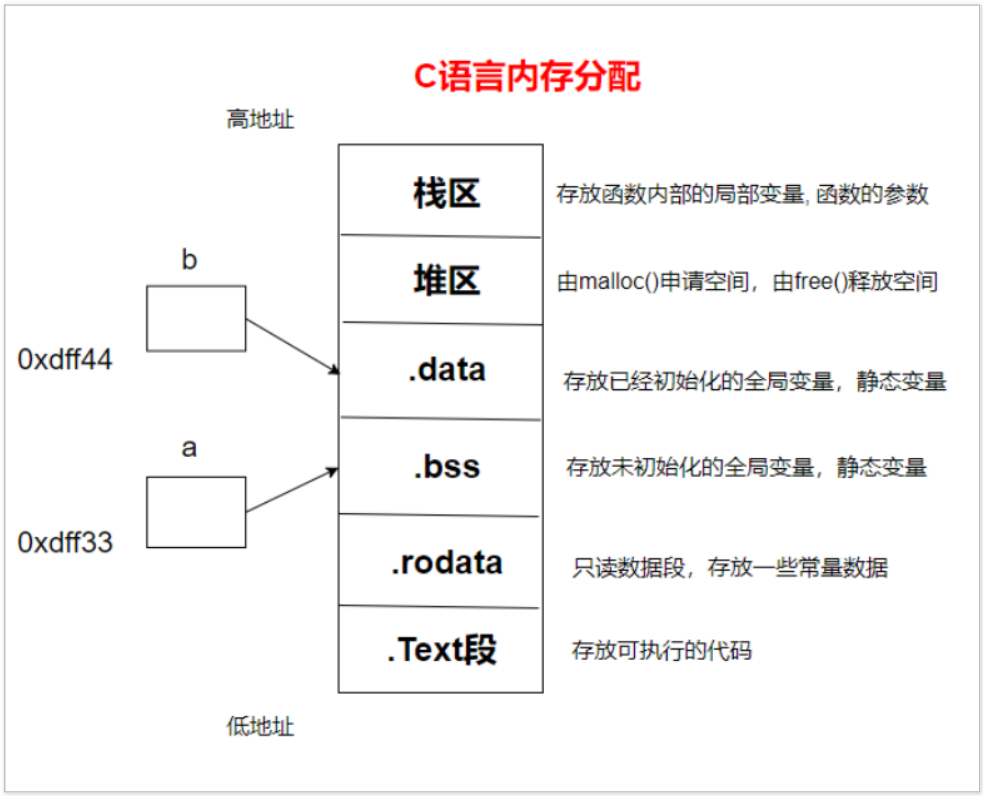
1. 在编译的过程中，会在数据区为该变量开辟空间，并对其进行初始化，如果代码中未对其进 行初始化，则系统默认初始化为 0。
2. 用 static 修饰的局部变量，会延长局部变量的寿命，超出函数的生存期。

示例代码：

```
#include <stdio.h>

void fun1()
{
    static int a;

    printf("a = %d\n",a);
}

void fun2()
{
    static int b = 10;
    b++;
    printf("b = %d\n",b);
}

void fun3()
```

```
{
    fun1();
    printf("======================\n");
    fun2();
    fun2();
    fun2();
    return 0;
}
```

内存分配图：



运行结果：

1. static 修饰全局变量，在数据区域分配存储空间，未初始化编译器会自动初始化为 0.
2. static 修饰全局变量，限制全局变量的使用范围，让其只能在本文件使用，其他文件不能使用。

globa.c

```
#include <stdio.h>

static int global_var = 10;
```

main.c

```
#include <stdio.h>

extern int global_var;

void fun()
{
    printf("global_var = %d\n", global_var);
    return;
}
```

函数的使用方式与全局变量类似，在函数的返回类型前加上 static，就是静态函数。其特性如下：

1. 静态函数只能在声明它的文件中可见，其他文件不能引用该函数

2. 不同的文件可以使用相同名字的静态函数，互不影响

fun1.c

```
#include <stdio.h>
static void fun(void)
{
    printf("hello from fun.\n");
}
int main(void)
{
    fun();
    fun1();
    return 0;
}
```

fun2.c

```
#include <stdio.h>
static void fun1(void)
{
    printf("hello from static fun1.\n");
}
```

编译结果:

```
static_fun.c:(.text+0x20)：对‘fun1’未定义的引用
collect2: error: ld returned 1 exit status
```

自己思考一下以下程序的结果。

```
#include <stdio.h>

void fun1(void)
{
    int n = 10;

    printf("n = %d\n", n);
    n++;
    printf("n++ = %d\n", n);
}
void fun2(void)
{
    static int n = 10;

    printf("static n = %d\n", n);
    n++;
    printf("n++ = %d\n", n);
}

int main()
{
    fun1();
    fun2();
    fun1();
    fun2();
    return 0;
}
```