

第五十四节： 从全局变量和局部变量中感悟“栈”为何物。

【54.1 本节阅读前的名词约定。】

变量可以粗略的分成两类，一类是全局变量，一类是局部变量。如果更深一步精细划分，全局变量还可以分成“普通全局变量”和“静态全局变量”，局部变量也可以分成“普通局部变量”和“静态局部变量”，也就是说，若精细划分，可以分成四类。其中“静态全局变量”和“静态局部变量”多了一个前缀“静态”，这个前缀“静态”是因为在普通的变量前面多加了一个修饰关键词“static”，这部分的内容后续章节会讲到。本节重点为了让大家理解内存模型的“栈”，暂时不考虑“静态变量”的情况，人为约定，本节所涉及的“全局变量”仅仅默认为“普通全局变量”，“局部变量”仅仅默认为“普通局部变量”。

【54.2 如何判定全局变量和局部变量？】

全局变量就是在函数外面定义的变量，局部变量就是在函数内部定义的变量，这是最直观的判定方法。下面的例子能很清晰地说明全局变量和局部变量的判定方法：

```
unsigned char a;    //在函数外面定义的，所以是全局变量。
void main()  //主函数
{
    unsigned char b; //在函数内部定义的，所以是局部变量。
    b=a;
    while(1)
    {

    }
}
```

【54.3 全局变量和局部变量的内存模型。】

单片机内存包括 ROM 和 RAM 两部分，ROM 存储的是单片机程序中的指令和一些不可更改的常量数据，而 RAM 存放的是可以被更改的变量数据，也就是说，全局变量和局部变量都是存放在 RAM，但是，虽然都是存放在 RAM，全局变量和局部变量之间的内存模型还是有明显的区别的，因此，分了两个不同的 RAM 区，全局变量占用的 RAM 区称为“全局数据区”，局部变量占用的 RAM 区称为“栈”，因为我后面会用宾馆来比喻“栈”，为了方便记忆，大家可以把“栈”想象成“客栈”来记忆。它们的内存模型到底有什么本质的区别呢？“全局数据区”就像你自己家的房间，是唯一的，一个房间的地址只能你一个人住（假设你还没结婚的时候），而且是永久的，所以说每个全局变量都有唯一对应的 RAM 地址，不可能重复的。而“栈”就像宾馆客栈，一年下来每天晚上住的人不一样，每个人在里面居住的时间是有期限的，不是长久的，一个房间的地址一年下来每天可能住进不同的人，不是唯一的。“全局数据区”的全局变量拥有永久产权，“栈”区的局部变量只能临时居住在宾馆客栈，地址不是唯一的，有期限的。全局变量像私人区，局部变量像公共区。“栈”的这片公共区，是给程序里所有函数内部的局部变量共用的，函数被调用的时候，该函数内部的每个局部变量就会被分配对应到“栈”的某个 RAM 地址，函数调用结束后，该局部变量就失效，因此它对应的“栈”的 RAM 空间就被收回以便给下一个被调用的函数的局部变量占用。请看下面这个例子，我借用“宾馆客栈”来比喻局部变量所在的“栈”。

```
void HanShu(void); //子函数的声明
```

```

void HanShu(void)    //子函数的定义
{
    unsigned char a;    //局部变量
    a=1;
}
void main() //主函数
{
    HanShu() ;        //子函数的调用
}

```

分析：上述例子，单片机从主函数 main 往下执行，首先遇到 HanShu 子函数的调用，所以就跳到 HanShu 函数的定义那里开始执行，此时的局部变量 a 开始被分配在 RAM 的“栈区”的某个地址，相当于你入住宾馆被分配到某个房间。单片机执行完子函数 HanShu 后，局部变量 a 在 RAM 的“栈区”所分配的地址被收回，局部变量 a 消失，被收回的 RAM 地址可能会被系统重新分配给其它被调用的函数的局部变量，此时相当于你离开宾馆，从此你跟那个宾馆的房间没有啥关系，你原来在宾馆入住的那个房间会被宾馆老板重新分配给其他的客人入住。全局变量的作用域是永久性不受范围限制的，而局部变量的作用域就是它所在函数的内部范围。全局变量的“全局数据区”是永久的私人房子（这里的“永久”仅仅是举一个例子，别拿“70 年产权”来抬杠），局部变量的“栈”是临时居住的“客栈”。重要的事情说两遍，再次总结如下：

（1）每定义一个新的全局变量，就意味着多开销一个新的 RAM 内存。而每定义一个局部变量，只要在函数内部所定义的局部变量总数不超过单片机的“栈”区，此时的局部变量不开销新的 RAM 内存，因为局部变量是临时借用“栈”区的，使用后就还给“栈”，“栈”是公共区，可以重复利用，可以服务若干个不同的函数内部的局部变量。

（2）单片机每次进入执行函数时，局部变量都会被初始化改变，而全局变量则不会被初始化，全局变量是一直保存之前最后一次更改的值。

【54.4 三个常见疑问。】

第一个疑问：

问：“全局数据区”和“栈区”是谁在幕后分配的，怎么分配的？

答：是 C 编译器自动分配的，至于怎么分配，谁分配多一点，谁分配少一点，C 编译器会有一个默认的比例分配，我们一般都不用管。

第二个疑问：

问：“栈”区是临时借用的，子函数被调用的时候，它内部的局部变量才会“临时”被分配到“栈”区的某个地址，那么问题来了，谁在幕后主持“栈区”这些分配的工作，难道也是 C 编译器？C 编译器不是在编译程序的时候一次性就做完了编译工作然后就退出历史舞台了吗？难道我们程序已经在单片机内部运转的时候，编译器此时还在幕后指手画脚的起作用？

答：单片机已经上电开始运行程序的时候，编译器是不可能起作用的。所以，真相只有一个，“栈区”分配给函数内部局部变量的工作，确实是 C 编译器做的，唯一需要注意的地方是，它不是“现炒现卖”，而是在单片机上电前，C 编译器就把所有函数内部的局部变量的分配工作就规划好了，都指定了如果某个函数一旦被调用，该函数内部的哪个局部变量应该分到“栈区”的哪个地址，C 编译器都是事先把这些“后事”都交代完毕了才“结束自己的生命”，后面，等单片机上电开始工作的时候，虽然 C 编译器此时“不在”了，但是单片机都是严格按照 C 编译器交代的“遗嘱”开始工作和分配“栈区”的。因此，“栈区”的“临时分配”非真正严格意义上的“临时分配”。

第三个疑问：

问：函数内部所定义的局部变量总数不超过单片机的“栈”区的 RAM 数量，那，万一超过了“栈”区的 RAM 数量，后果严重吗？

答：后果特别严重。这种情况，专业术语叫“爆栈”。程序会出现异常，而且是莫名其妙的异常。为了避免这种情况，一般在编写程序的时候，函数内部都不能定义大数组的局部变量，局部变量的数量不能定义太多太大，尤其要避免刚才所说的定义开辟大数组局部变量这种情况。大数组的定义应该定义成全局变量，或者定义成“静态的局部变量”（“静态”这部分相关的内容后面章节会讲到）。有一些 C 编译器，遇到“爆栈”的情况，会好心跟你提醒让你编译不过去，但是也有一些 C 编译器可能就不会给你提醒，所以大家以后做项目写函数的时候，要对“爆栈”心存敬畏。

【54.5 全局变量和局部变量的优先级。】

刚才说到，全局变量的作用域是永久性并且不受范围限制的，而局部变量的作用域就是它所在函数的内部范围，那么问题来了，假如局部变量和全局变量的名字重名了，此时函数内部执行的变量到底是局部变量还是全局变量？这个问题就涉及到优先级。注意，当面对同名的局部变量和全局变量时，函数内部执行的变量是局部变量，也就是局部变量在函数内部要比全局变量的优先级高。为了深刻理解“全局变量和局部变量的优先级”，强烈建议大家必须仔细看完下面列举的三个练习例子。

【54.6 例程练习和分析。】

请看下面第一个例子：

```
/*---C 语言学习区域的开始。-----*/

unsigned char a=5;      //此处第 1 个 a 是全局变量。

void main() //主函数
{
    unsigned char a=2; //此处第 2 个 a 是局部变量。跟上面全局变量的第 1 个 a 重名了！

    View(a); //把 a 发送到电脑端的串口助手软件上观察。
    while(1)
    {

    }

}

/*---C 语言学习区域的结束。-----*/
```

分析：

上述例子，有 2 个变量重名了！其中一个是全局变量，另外一个局部变量。此时输出显示的结果是 5 还是 2？正确的答案是 2。因为在函数内部，函数内部的局部变量比全局变量的优先级更加高。此时 View(a) 是第 2 个局部变量的 a，而不是第 1 个全局变量的 a。虽然这里的两个 a 重名了，但是它们的内存模型不一样，第 1 个全局变量的 a 是分配在“全局数据区”是具有唯一的地址的，而第 2 个局部变量的 a 是被分配在

临时的“栈”区的，寄生在 main 函数内部。

再看下面第二个例子：

```
/*---C 语言学习区域的开始。-----*/
void HanShu(void); //函数声明
unsigned char a=5;    //此处第 1 个 a 是全局变量。
void HanShu(void)    //函数定义
{
    unsigned char a=3; //此处第 2 个 a 是局部变量。
}
void main() //主函数
{
    unsigned char a=2; //此处第 3 个 a 也是局部变量。
    HanShu(); //子函数被调用
    View(a); //把 a 发送到电脑端的串口助手软件上观察。
    while(1)
    {

    }

}
/*---C 语言学习区域的结束。-----*/
```

分析：

上述例子，有 3 个变量重名了！其中一个是全局变量，另外两个是局部变量。此时输出显示的结果是 5 还是 3 还是 2？正确的答案是 2。因为，HanShu 这个子函数是被调用结束之后，才执行 View(a) 的，就意味 HanShu 函数内部的局部变量(第 2 个局部变量 a)是在执行 View(a) 语句的时候就消亡不存在了，所以此时 View(a) 的 a 是第 3 个局部变量的 a（在 main 函数内部定义的局部变量的 a）。

再看下面第三个例子：

```
/*---C 语言学习区域的开始。-----*/
void HanShu(void); //函数声明
unsigned char a=5;    //此处第 1 个 a 是全局变量。
void HanShu(void)    //函数定义
{
    unsigned char a=3; //此处第 2 个 a 是局部变量。
}
void main() //主函数
{
    HanShu(); //子函数被调用
    View(a); //把 a 发送到电脑端的串口助手软件上观察。
}
```

```
while(1)
{

}

}

/*---C 语言学习区域的结束。-----*/
```

分析：

上述例子，有 2 个变量重名了！其中一个是全局变量，另外一个局部变量。此时输出显示的结果是 5 还是 3？正确的答案是 5。因为，HanShu 这个子函数是被调用结束之后，才执行 View(a) 的，就意味 HanShu 函数内部的局部变量(第 2 个局部变量)是在执行 View(a) 语句的时候就消亡不存在了，同时，因为此时 main 函数内部也没有定义 a 的局部变量，所以此时 View(a) 的 a 是必然只能是第 1 个全局变量的 a（在 main 函数外面定义的全局变量的 a）。

【54.7 如何在单片机上练习本章节 C 语言程序？】

直接复制前面章节中第十一节的模板程序，练习代码时只需要更改“C 语言学习区域”的代码就可以了，其它部分的代码不要动。编译后，把程序下载进带串口的 51 学习板，通过电脑端的串口助手软件就可以观察到不同的变量数值，详细方法请看第十一节内容。